INF

BULETIN AL CLUBULUI PROGRAMATORILOR

INF

BULETIN
al
CLUBULUI
PROGRAMATORILOR

MINISTERUL
EDUCAȚIEI ȘI
ÎNVĂȚĂMÎNTULUI
Casa Universitarilor
Timișoara

Buletin al
Clubului Programatorilor

INF

Nr. 1/1989

COLECTIVUL DE REDACȚIE:

conf. dr. ing. CRIȘAN STRUGARU
ș.l. dr. ing. ȘTEFAN HOLBAN
ș.l. dr. ing. IONEL JIAN
ing. CONSTANTIN COZMIUC

TEHNOREDACTAREA

EWELINE BELMUSTAȚĂ
CRISTIAN BÎRLONCEA

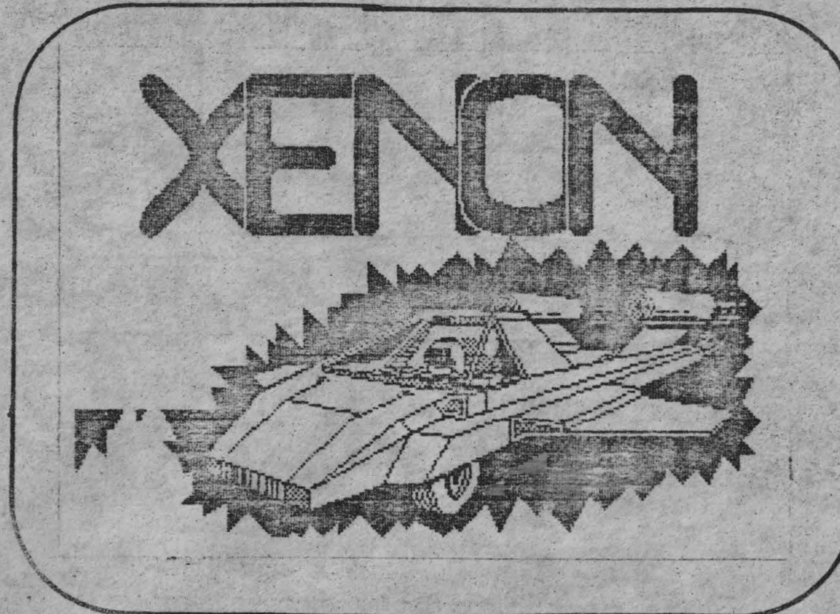
SUMAR

CALCULATORUL IN SPRIJINUL DUMNEAVOASTRĂ

MANUAL DE UTILIZARE

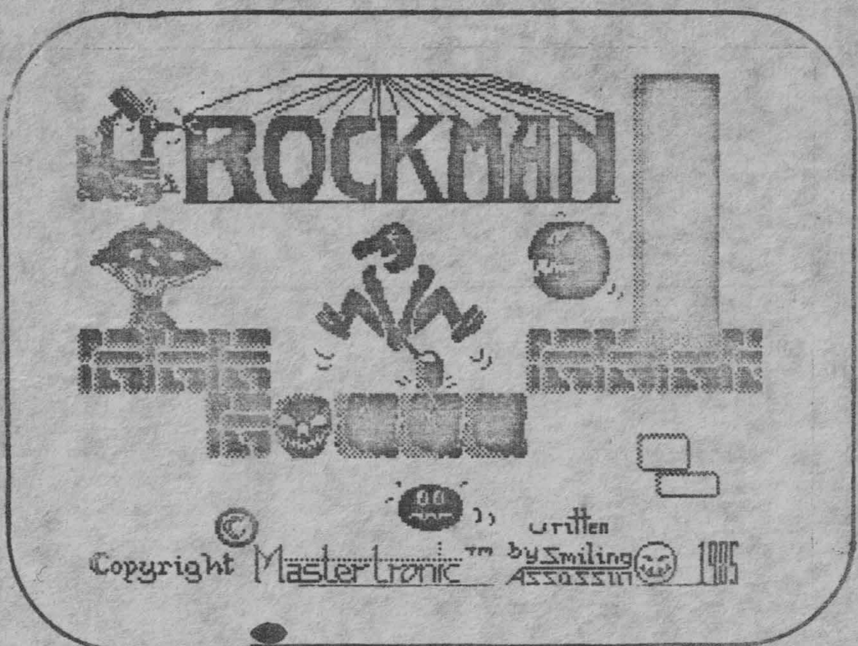
PROGRAME

DIVERSE



INSEMNĂRILE UNOR AVENTURIERI

• ING TIBERIU ONU
• ING MIODRAG PUTERITY



Unii le numesc scenarii interactive. Altii , refeferindu-se la ele , folosesc termenul de jocuri de rol. Dar denumirea lor consacrata este cea de AVENTURI (adventures). Au aparut odata cu explozia microcalculatoarelor , atunci :ind in salile de jocuri video , jucatorii cu nervi de otel si eflexe sincronizate la microsecunda se incapatinau sa manevreze joystick-urile consolelor de joc , incapabile de a suporta dezvoltarea unui alt tip de software decit jocurile de actiune.

Se poate spune , pe drept cuvint , ca aventurile au provocat o revolutie in domeniul jocurilor computerizate , adaugind o noua dimensiune software-ului de divertisment.

Dar ce reprezinta de fapt aceste aventuri ?

Pentru a nu fi nevoiti sa dam o defihitie rigida , sa ne imaginam un experiment de parapsihologie :

Sa presupunem ca subiectul experimentului doreste sa ia parte la o actiune care are loc in alt univers decit universul propriu - de exemplu intr-un univers decalat temporar sau intr-o lume paralela. Evident , experienta subiectului intr-un astfel de univers , reprezinta o cumulare de informatii si senzatii legate de universul cu pricina. Cu alte cuvinte , subiectul nu va participa niciodata fizic la actiunile care se petrec in universul destinatie , ci isi va imagina doar acest lucru. In fine , subiectul nu va putea nici macar sa perceapa universul destinatie , decit in prezenta unui mediu (persoana releu).

In cazul aventurilor , jucatorul este subiectul unui experiment similar. Universul destinatie este implementat soft , iar calculatorul joaca rolul de mediu. Din pacate , calculatorul nu poate sa transmita senzatii (in sensul propriu al cuvintului) asa ca experienta unui jucator de aventuri se va rezuma doar la un schimb de informatii privind universul implementat si actiu-

nile care se eruleaza in el. Este de la sine inteles ca proprietatile de mediu ale calculatorului , adica multitudinea informatiilor pe care acesta le poate vehicula , depind puternic de memoria disponibila.

Vom fi poate criticati ca intr-un buletin ca si INF sfera jocurilor nu isi are locul. Anticipind astfel de critici , raspundem ca intentia noastra nu este nici pe departe aceea de a discuta doar pe marginea jocurilor computerizate. Aventurile reprezinta insa dezvoltari software deosebit de pretentioase care necesita tehnici de virf in programare. Daca doriti , puteti sa considerati ca vom folosi aventurile ca un pretext agreabil pentru a descrie unele din aceste tehnici expert , carora sintem siguri ca le veti gasi aplicarea atit in sfera ludicului cit si in sfera utilitara.

Lucrarea de fata se adreseaza tuturor acelora care poseda cunostinte elementare in programare. Exemplificarile au fost realizate pentru calculatoare compatibile SPECTRUM si ne exprimam convingerea ca dupa intelegerea algoritmilor care stau la baza programelor si subrutinelor prezentate , eventualele conversii nu vor ridica probleme. In unele cazuri au fost prezentate si exemple in limbaj BASIC , desi noi optam pentru limbajul de asamblare , din motive lesne de inteles.

Dar sa incepem acum cu

UN SCURT ISTORIC

Primele aventuri aparute pe piata SINCLAIR erau simpliste din punct de vedere tehnic si in general neatractive. Este vorba de aventurile pur textuale , la care calculatorul descria in cuvinte diverse locatii si actiuni dupa care tiparea prompterul stereotip :

Tell me what to do >
(Spune-mi ce sa fac >)

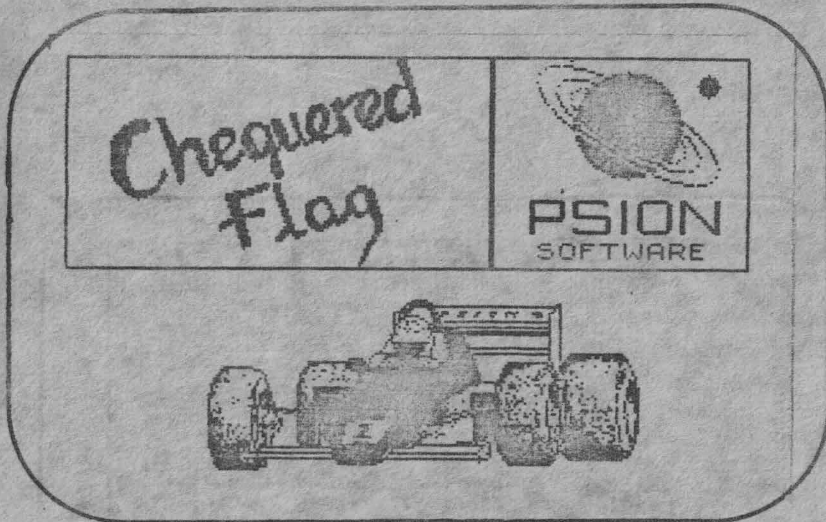
Jucatorul introducea comenzi in limbajul natural , in forma impusa VERB - SUBSTANTIV , dar datorita vocabularului redus, putine dintre acestea erau recunoscute de catre interpretorul de sintaxa , iar comenzile mai complicate erau pur si simplu ignorate (Pardon ?).

Treptat , treptat , tehnicile de programare au fost imbunatatite , vocabularele au devenit din ce in ce mai cuprinzatoare iar locatiile din ce in ce mai numeroase. Au aparut aventurile grafice , la care descrierile textuale erau completate de afisarea (uneori optionala) a unor imagini sintetice.

Un nou drum a fost deschis de catre aventurile inteligente (THE HOBBIT , SHERLOCK , ID , THE PAWN) , despre care vom vorbi ceva mai tirziu.

Interpretoarele si corectoarele de sintaxa au fost perfec-





tionate , remarcabil fiind interpretorul lui THE PAWN , o bijuterie soft care permite evaluarea unor propuzitii cum ar fi :

Throw the rope over the chasm , then pull it firmly
(Arunca fringhia peste prapastie , apoi trage de ea cu hotarire)

Amintim insa ca in cazul acestui program s-a renuntat la descrierea grafica , din lipsa de memorie.

Unii programatori vizind eliberarea de vocabular au inceput sa utilizeze comanda prin meniuri (Melbourne House-DOC THE DESTROYER sau si mai reprezentativ sistemul WINDIMATION al firmei Mastertronics , utilizat in gama MAD) sau prin iconograme (Electronic Pencil-THE FOURTH PROTOCOL).

Raspunzind dorintei de a imbina elementele de aventura cu cele proprii jocurilor de actiune , a aparut chiar o categorie noua de jocuri , asa numitele arcade adventures (aventuri cu element de actiune).

Majoritatea aventurilor care circula la noi in tara sint de origine britanica , fiind total inaccesibile celor ce nu cunosc limba engleza si greu accesibile celor ce nu cunosc aceasta limba la perfectie. Din acest motiv , anuntam pe aceasta cale ca odata cu cea de-a doua parte a articolului nostru , vom pune la dispozitie si prima aventura in limba romana.

SIMPLU CA BUNA ZIUA !

Dupa cum credem ca ati inteles deja , scrierea aventurilor implica doua activitati complet diferite in esenta : punerea in scena si programarea - si in acest sens , amintim ca marile firme de software lucreaza cu persoane specializate in scriere de scenarii , grafica pe calculator , programare , muzica digitala si respectiv emulare (in vederea conversiilor).

Este drept ca BASIC este un limbaj familiar majoritatii celor care poseda calculatoare personale , dar un utilizator care ar dori sa dezvolte o aventura in acest limbaj , dupa depunerea unui efort deloc neglijabil , ar ramine cu siguranta dezamagit constatind ca produsul sau final este un program lent si care consuma in mod inutil o cantitate insemnata de memorie (sa nu uitam ca un indicator de conditie ocupa 5 octeti in BASIC si un singur bit in limbaj masina). Ce diferenta fata de aventurile profesionale , care sint suple si incoparabil mai rapide !

Se pare deci ca unui scenarist talentat , dar care nu exceleaza in arta programarii nu ii ramine altceva de facut decit sa renunte la ideea de a scrie de unul singur o aventura competitiva. Ei bine , nu ! Firmele de software s-au gindit si la aceasta.

Astfel au aparut programele specializate in scrierea aventurilor , dintre care mai demne de luat in seama sint THE QUILL si GRAPHIC ADVENTURE CREATOR (G.A.C.). Reclamele declara ca folosind aceste produse , pot fi scrise aventuri in cod masina simplu ca buna ziua. Dupa cum veti vedea insa , utilizarea eficienta a acestor programe necesita totusi stapinirea unor notiuni si artificii de programare de loc comune.

Vom face in continuare o scurta descriere a lui G.A.C. , cu specificarea ca THE QUILL , mai simplist si greoi in utilizare , manipuleaza notiuni asemanatoare.

G.A.C. este de fapt un compilator al unui metalimbaj specializat in scrierea aventurilor. Fisierul sursa al acestui compilator reprezinta o descriere a entitatilor care intervin in aventura si a modului in care sint tratate acestea. Fisierul obiect va fi , desigur , o aventura rulabila in cod masina.

Entitatile care intervin in aventura sint :

- ENTITATI SINTACTICE - substantive
  - adverbe
  - verbe
- MESAJE
- OBIECTE
- LOCATII

Fiecare dintre aceste entitati este codificata numeric conform unei corespondente definite de utilizator.

Prima faza a programarii unei aventuri in G.A.C. (se presupune ca scenariul exista deja) , o reprezinta chiar definirea entitatilor si a modului de codificare a lor. Si , de regula , se incepe cu entitatile sintactice :

[A]dverbes - adverbe
[N]ouns - substantive
[V]erbes - verbe

Modul de definire a entitatilor este oarecum similar introducerii liniilor de program BASIC. Se introduce numarul de ordine al entitatii , urmat de un spatiu (separator) si de entitatea in sine , definita la nivel alfanumeric.

Gasindu-ne de exemplu la capitolul [N]ouns , in urma definitiei :

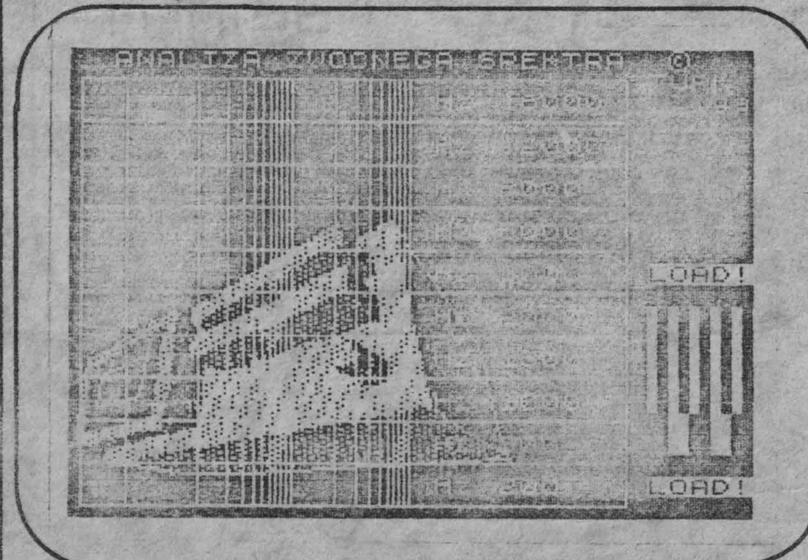
5 SABIE

vom obtine substantivul SABIE , cu numarul de ordine 5 conform codificarii dorite. In continuare , referirile la acest substantiv vor fi facute prin intermediul numarului sau de ordine. Problema sinonimelor se rezolva asociind doua descrieri alfanumerice aceluiasi numar de ordine. Aplicatia COLECTIE DE SUBSTANTIVE -> CODURI nu va fi deci in mod neaparat injectiva. Definind :

5 SPADA

vom obtine un acelasi efect introducind urmatoarele doua comenzi

IA SPADA
IA SABIA



Aceleasi consideratii sint valabile atit pentru adverbe cit si pentru verbe.

In urma definirii entitatilor sintactice se obtine vocabularul aventurii. Interpretorul de sintaxa nu va recunoaste cuvinte care nu fac parte din acest vocabular.

Mesajele se codifica in mod analog , cu diferenta ca ele reprezinta de fapt propozitii sau paragrafe si ca nu sint evaluate niciodata , fiind folosite doar la afisare.

Obiectele reprezinta entitati de sine statatoare. Ele nu trebuie sa fie confundate cu substantivele. Substantivele intervin doar in analiza sintactica. Obiectele sint , daca vreti , imaginea implementata a entitatilor descrise prin substantive. Un obiect este caracterizat printr-o descriere , o constanta numerica , reprezentind greutatea obiectului si o serie de indicatori de conditie predefiniti. Acesti indicatori de conditie stabilesc daca obiectul este sau nu prezent , disponibil sau face parte din inventarul eroului. In faza de initializare se stabileste si numarul camerei in care obiectul cu pricina se gaseste la inceputul unui nou joc. Un obiect inexistent se considera a fi initial in camera 0 , camera care de fapt nu exista. Sa mai punctam aici o idee : o luminare stinsa reprezinta un cu totul alt obiect decit aceeasi luminare aprinsa. Aceasta deoarece parametrii unui obiect pot varia la trecerea sa dintr-o stare in alta si cu siguranta se va modifica si descrierea aferenta. La aprinderea luminarii va trebui asadar sa distrugem luminarea stinsa (mutind-o in locatia 0 = obiect inexistent) si sa creem un obiect nou : luminarea aprinsa.

Locatiile sau camerele ([R]ooms) reprezinta teatrul de actiune al aventurii. Fiecare camera este determinata de o descriere (analog mesajelor) si de o lista de conexiuni. Lista de conexiuni determina harta aventurii si trebuie construita in mod logic (este absurd ca urcind o scara sa ajungem din camera 1 in camera 2 si continuind urcusul sa ajungem din nou in camera 1 - bug sesizat si in aventuri 'serioase' - vezi THE CODE). Aceasta lista de conexiuni se defineste ca o succesiune de verbe si numere de locatii , separate prin spatii , cu specificarea ca atit verbele cit si locatiile cu pricina trebuie sa fie descrise la optiunile corespunzatoare. Sa consideram un exemplu (fig.1):

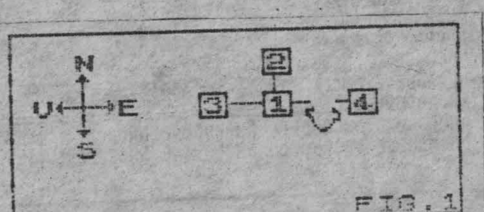


FIG.1

Lista de conexiuni pentru locatia 1 va fi :

NORD 2 VEST 3 SARI 4

iar pentru locatia 2 :

SUD 1

Se recomanda celui care scrie scenariul sa deseneze initial o astfel de harta pe hirtie.

In continuare vom descrie metalimbajul G.A.C. , un exemplu excelent de dezvoltare a unui sistem dedicat acestei laturi a inteligentei artificiale. Dar inainte de aceasta sa studiem putin modul in care are loc tratarea in faza de executie a aventurilor G.A.C. (fig.2):

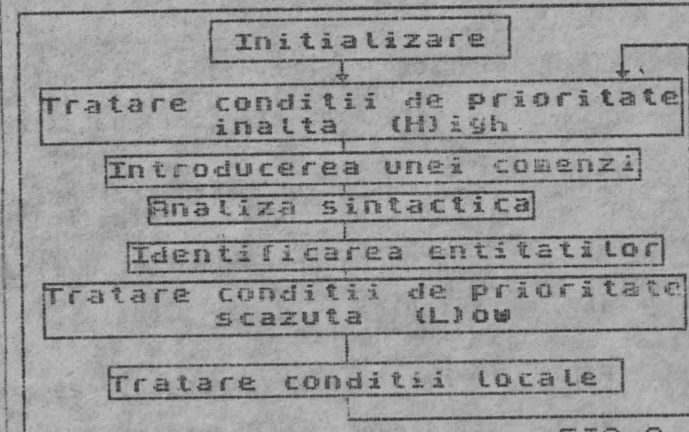


FIG.2

Se observa ca tratarea conditiilor se face pe trei nivele de prioritate : inalta , scazuta si locala si nu este lipsit de importanta faptul ca prima tratare are loc inainte de introducerea comenzilor de la tastatura.

Pentru fiecare dintre cele 3 nivele , se elaboreaza linii de program care vor fi executate secvential.

Metalimbajul G.A.C. cuprinde comenzi , functii , operatori logici si relationali si variabile ale utilizatorului , toate proprii.

COMENZILE

IF (exp) lista_de_comenzi - alternativa conditionala comuna majoritatii limbajelor de programare. In cazul in care expresia booleana da rezultatul 'fals' , se trece la tratarea urmatoarei linii.

END - produce inhibarea tratarii curente. Tratarea se continua de la nivelul imediat inferior.

LF - produce transmiterea unui caracter de line-feed.

TEXT - inhiba afisarea imaginilor create la optiunea [G]raphics.

PICT - reactualizeaza descrierea grafica.

WAIT - asteapta introducerea de la tastatura a unei noi comenzi.

HOLD (x)- produce o intirziere egala ca si durata cu x cadre TV.

SAVE - salveaza baza de date a aventurii (status) , pentru a putea fi continuata din aceeasi pozitie.

LOAD - incarca baza de date a aventurii.

PRIN (x)- produce scrierea valorii numerice x.

LIST (x)- produce afisarea listei x.

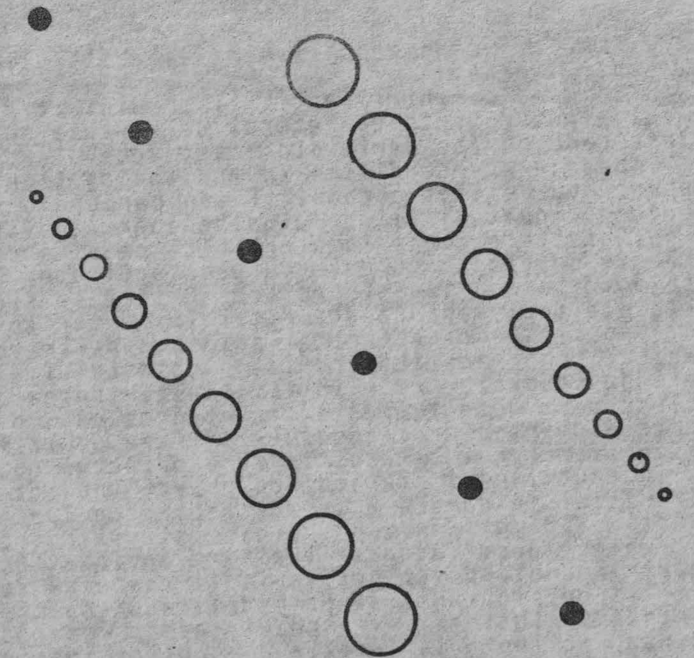
MESS (x)- produce afisarea mesajului cu numarul de ordine x

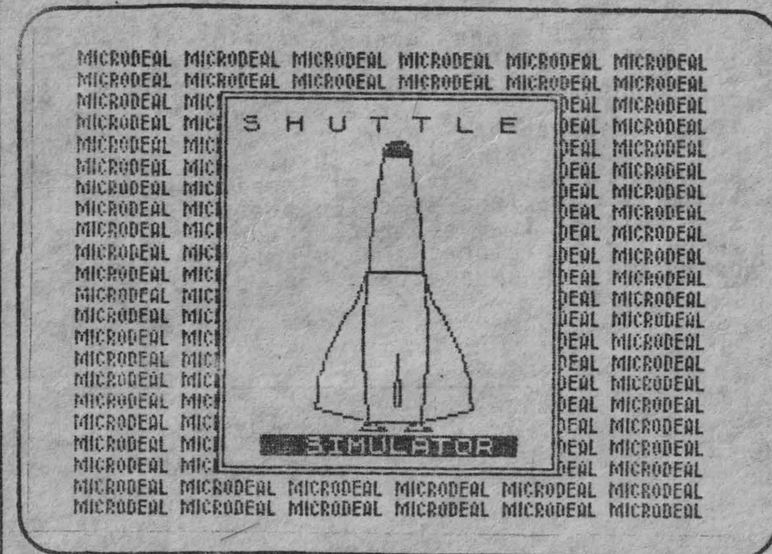
DESC (x)- produce afisarea descrierii camerei x.

LOOK - produce descrierea locatiei curente (echivalent cu DESC (ROOM).

GET (x) - produce setarea indicatorului de inventar pentru obiectul numarul x. Acest obiect va fi la purtator.

DROP (x)- produce resetarea indicatorului de inventar pen-





tru obiectul numarul x. Acest obiect nu va mai fi in posesia eroului.

x SWAP y- schimba obiectele x si y intre ele.

GOTO (x)- face ca locatia curenta sa fie locatia x.

FIND (x)- face ca locatia curenta sa fie locatia in care se gaseste obiectul cu numarul x.

BRIN (x)- aduce obiectul numarul x in locatia curenta.

x TO y - duce obiectul numarul x in locatia y. Utilizata frecvent pentru a distruge obiectele prin x TO 0.

STRE (x)- stabileste forta eroului (greutatea maxima pe care o poate cara) la valoarea expresiei x.

SET (x) - seteaza indicatorul de conditie cu numarul x.

RESE (x)- reseteaza indicatorul de conditie cu numarul x.

x CSET y- asigneaza variabilei y valoarea expresiei x.

DECR (x)- decrementeaza valoarea variabilei cu numarul x.

INCR (x)- incrementeaza valoarea variabilei cu numarul x.

OKAY - trimite spre ecran mesajul 254 (predefinit : Okay dar poate fi modificat). Echivalent cu MESS (254)

EXIT - incheie aventura. Afiseaza mesajul 243 (predefinit : Press a key for another game) , asteapta apasarea unei taste si reincepe un joc nou.

QUIT - afiseaza mesajul 244 (predefinit : Are you sure ? (Y/N)). Daca se apasa tasta N se revine la joc. Daca se apasa Y se executa EXIT.

FUNCTII

OBJ (x) - are ca rezultat descrierea obiectului x.

RAND (x)- are ca rezultat un numar aleator din plaja 0..x.

SET? (x)- functie booleana cu rezultatul adevarat daca indicatorul de conditie numarul x este setat.

RES? (x)- functie booleana cu rezultatul adevarat daca indicatorul de conditie numarul x este resetat.

x EQU? y- functie booleana cu rezultatul adevarat daca variabila numarul y are aceeasi valoare ca si expresia de test x.

ROOM - da ca rezultat numarul locatiei curente.

WITH - da ca rezultat lista obiectelor din inventar.

WEIG (x)- da ca rezultat greutatea obiectului cu numarul x.

CONN (x)- da ca rezultat numarul locatiei cu care locatia curenta are conexiune (conform hartii) prin intermediul verbului cu numarul x. In cazul in care locatia curenta nu are legaturi cu alte camere prin verbul specificat , intoarce rezultatul 0. Pentru exemplul anterior , gasindu-ne in locatia 1 si avind in baza de date a verbelor valorile asociate :

```
NORD 1
SUD  13
EST  200
VEST 27
SARI 50
```

vom avea :

```
CONN (1) = 2
CONN (50) = 4
CONN (13) = 0
```

AT (x) - functie booleana cu rezultatul adevarat daca locatia curenta este locatia numarul x.

HERE (x)- functie booleana cu rezultatul adevarat daca obiectul numarul x este prezent in locatia curenta

CARR (x)- functie booleana cu rezultatul adevarat daca obiectul x face parte din inventarul eroului.

AVAI (x)- functie booleana cu rezultatul adevarat daca obiectul numarul x este disponibil. Echivalenta cu HERE (x) OR CARR (x).

TURN - are ca rezultat numarul actiunilor executate. Utila in determinarea procentajelor de eficienta.

NOUN (x)- functie booleana cu valoarea adevarat daca ultimul substantiv introdus a fost substantivul cu numarul x.

VERB (x)- analog pentru verbe.

ADVE (x)- analog pentru adverbe.

VBNO - are ca rezultat numarul de ordine al ultimului

verb introdus. In cazul in care verbul cu pricina nu este gasit in vocabular , intoarce valoarea 0.

NO1,NO2 - analog pentru substantivul 1 si substantivul 2. Analizorul de sintaxa deceleaza doua substantive pentru a putea detecta entitatile sintactice ale unor propozitii de forma: DA-I VRAJITORULUI SABIA

OPERATORI LOGICI

AND - SI logic

OR - SAU logic

NOT - negatie

XOR - SAU EXCLUSIV

OPERATORI RELATIONALI

< , > , = , <= , >= , <>

CONTOARE

Sint disponibile 127 de variabile contor , adresabile prin CTR (x) si care pot suferi cresteri sau scaderi. De exemplu :

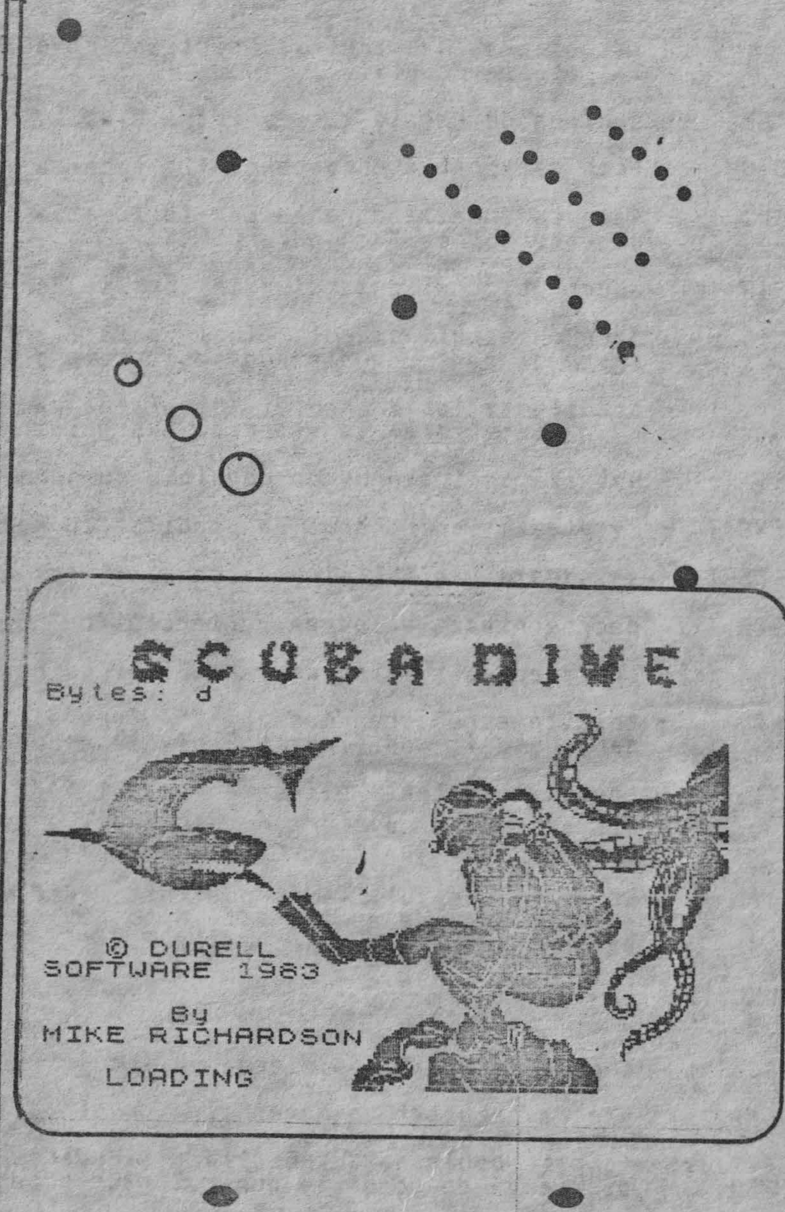
CTR (5) + 15 sau
CTR (7) - 3

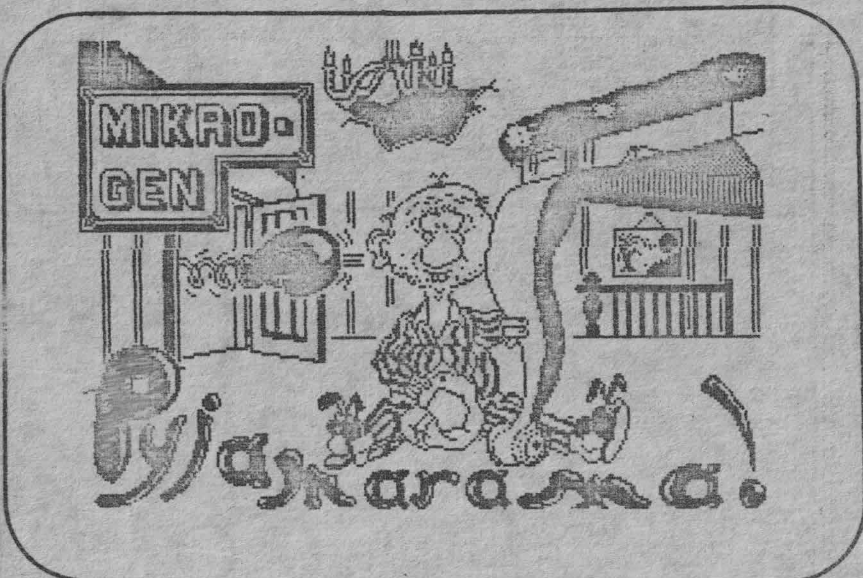
CTR (Ø) este rezervat procentajului de eficienta , iar CTR (126) numarului de actiuni care au avut loc in aventura.

Toti indicatorii de conditie si toate variabilele sint initializate la valori nule.

Optiunea [G]raphics permite definirea unor imagini grafice. Iata in continuare si comenzile disponibile in cadrul acestei optiuni :

- 5,6,7,8 - miscarea cursorului (impreuna cu CAPS - rapid)
- P - PAPER (Ø..9)
- I - INK (Ø..9)
- T - BORDER (Ø..7)
- B - BRIGHT (Ø,1 sau 8)
- V - FLASH (Ø,1 sau 8)
- C - marcheaza pozitia cursorului prin aplicarea temporara a unui atribut in starea FLASH 1
- Z - sterge ecranul , dar editarea grafica se continua. (zero)
- G - rastru de BRIGHT. (grid)





M - amesteca o imagine cu o alta imagine. (merge)
W - repeta succesiunea operatiilor , pina la operatia curenta. (draw all)
S - fill rapid. (special fill)
A - insereaza atributul curent la pozitia cursorului. (attribute)
D - PLOT in modul OVER 1.
F - PLOT in modul OVER 0.
L - linii. (lines)
R - dreptunghiuri. (rectangles)
E - elipse. (ellipses)

L,R si E se executa in modul elastic (asemanator cu ART STUDIO).

Grafica obtinuta este simplista , dar ocupa foarte putina memorie , datorita faptului ca in baza de date grafica nu se retine imaginea in sine , ci algoritmul prin care ea a fost creata (dupa cum veti vedea si intr-unul din paragrafele urmatoare).

G.A.C. permite si testarea aventurilor , punind la dispozitie o modalitate de diagnosticare prin afisarea starii celor 255 de indicatori de conditie , respectiv 128 de variabile CTR.

Va sfatuim sa urmariti cu atentie continutul fisierului exemplu - ADVINMAN , pentru ca acesta va poate edifica asupra stilului de programare in G.A.C.

PRO SI CONTRA

De ce sa mai continuam discutia ? Exista asadar un program specializat cu ajutorul caruia putem sa scriem fara bataie de cap , aventuri in cod masina.

Dar...

Si totusi exista un "dar". In primul rind , de la o vreme incoace , autorii acestui articol va pot spune aruncind o singura privire daca o anume aventura a fost scrisa in QUILL , G.A.C. sau daca este o aventura originala. Intr-adevar , aventurile scrise in metalimbaje specializate sint scoase parca de sub acelasi sablon. Le lipseste originalitatea tehnica si conceptuala , ceea ce se resimte imediat in interesul fata de joc.

In al doilea rind , trecind peste latura tehnica a problemei , gasim ca aventurile pot reprezenta terenul unor incercari mai serioase de introducere a inteligentei artificiale. Iar obtinerea unei aventuri mai inteligente , care eventual sa se produca si in timp real , sa posede o grafica cu adevarat atractiva , sunet si chiar voce sintetica , nu este posibila decit prin generarea unui sistem propriu de operare. Asadar , noi am ales

MODALITATEA MAI DIFICILA

dar care ne va da in final satisfactii depline , fiindca vom obtine ceea ce ne dorim.

Scrierea unui sistem de operare nu este o joaca de copil. Va trebui sa uitam complet de primii 16 K de memorie (ROM) , de-

oarece in co tinuare nu vom folosi aproape nimic din ei.

Sa recapitulam : ne gasim in fata unui calculator cu care vrem sa comunicam prin introducere de comenzi de la tastatura si receptionare de mesaje coerente pe ecran , ca efect al comenzilor introduse.

Simplificind problema si ignorind deocamdata elementele de grafica si sunet , putem considera ca dialogul nostru se realizeaza la nivel alfanumeric. Ne vom referi deci in cele ce urmeaza la

PRELUCRAREA INFORMATIEI ALFANUMERICE

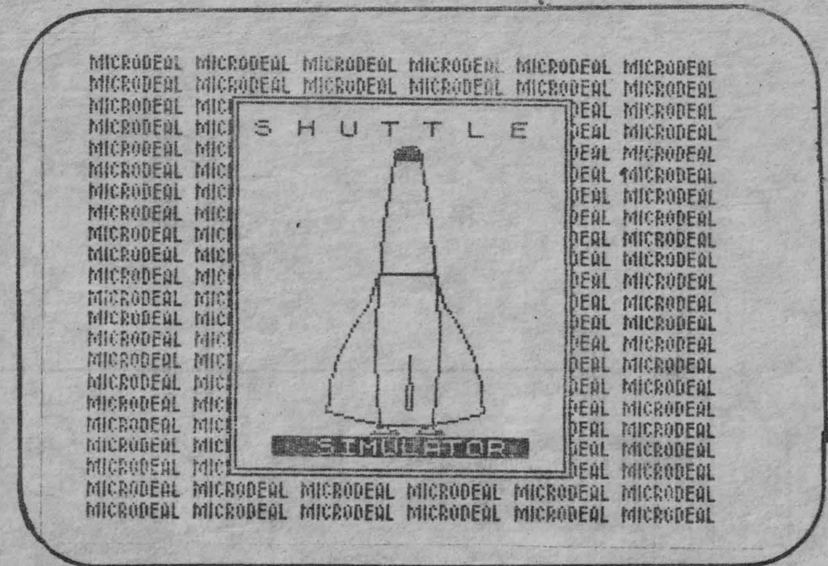
Calculatorul electronic este un instrument care prin natura sa lucreaza cu informatie binara. Pentru a putea manipula informatie alfanumerica (caractere , cifre si simboluri speciale) , vom avea nevoie deci de o codificare numerica a acestora , adica o aplicatie bijectiva care sa proiecteze multimea simbolurilor alfanumerice in domeniul valorilor numerice pe care calculatorul le poate prelucra. Este cunoscut faptul ca SPECTRUM lucreaza in jurul unui microprocesor Z80 , care poate efectua uzual operatii pe 8 biti de informatie. Acesti 8 biti permit realizarea a 2 ↑ 8 = 256 de combinatii din plaja zecimala [0..255]. Pe 8 biti vom putea retine asadar un simbol alfanumeric, dintr-o multitudine de 256 de simboluri.

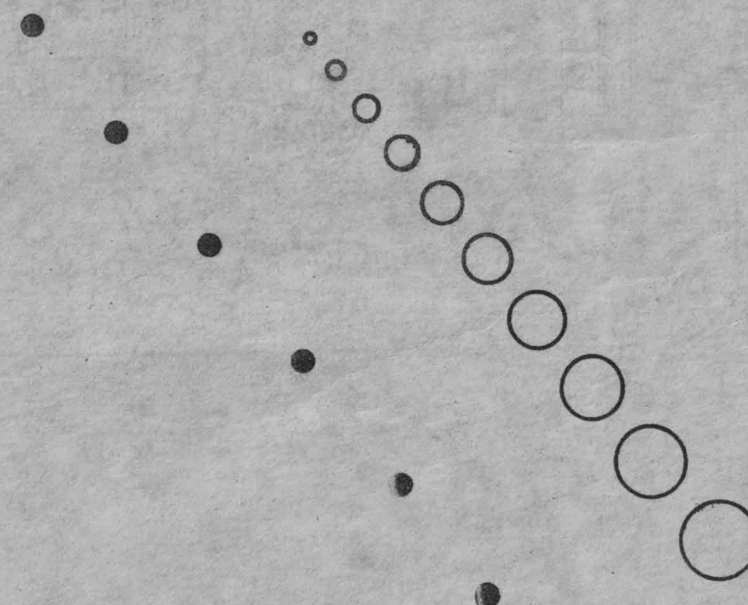
Eforturile pentru codificarea alfanumerica dateaza inca dinaintea aparitiei calculatoarelor electronice. Codurile alfanumerice MORSE si BAUDOT erau frecvent utilizate in teletransmisie. Ambele coduri foloseau pentru codificarea simbolurilor 5 biti de informatie , ceea ce permitea existenta unei colectii de 2 ↑ 5 = 32 caractere sau chiar 62 de caractere , in ideea folosirii unei anumite combinatii binare ca si caracter de schimbare a secventei.

Dupa aparitia primelor calculatoare au fost create codurile dedicate (Hollerith , EBCDIC) , dintre care s-a afirmat , mai ales in ultima perioada , ASCII (American Standard Code For Informations Interchange). Acest cod este utilizat si de sistemul de operare BASIC al calculatorului SPECTRUM si este prezentat in detaliu la sfirsitul manualului sau de utilizare.

Studiind acest cod , observam ca simbolurile propriu zise au valori corespondente intre 32 (SPACE) si 127 (@) , fiind deci in numar de 96. Valorile mai mici decit 32 corespund unor caractere de control speciale , iar valorile mai mari decit 127 , cuvintelor cheie ale sistemului de operare BASIC. Aceasta inseamna ca , simbolurile alfanumerice sint caracterizate in binar prin faptul ca au bitul 7 resetat (2 ↑ 7 = 128). Aceasta proprietate este speculata de unii programatori (printre care si maestrii de la Sinclair Research) , pentru a detecta sfirsitul unei entitati sintactice. Ultimului caracter al unei astfel de entitati i se seteaza bitul 7 , ceilalti biti fiind lasati nemodificati. La tiparire , calculatorul face urmatorul rationament :

1) Se ia un octet de la adresa curenta.





2) Daca bitul 7 al acestui octet este resetat , atunci se tipareste caracterul corespondent prin codul ASCII , se trece la urmatoarea adresa si se sare din nou la punctul 1).

3) Daca bitul 7 al octetului este setat , se reseteaza acest bit , se tipareste caracterul corespunzator noii valori si se incheie tiparirea.

Pe baza acestui rationament , autorii au creat un program destinat cautarilor ASCII. Aceasta este o facilitate comuna multor monitoare , dar atu-urile principle ale programului nostru sint faptul ca rezida in memoria video lasind libera toata zona de RAM incepind de la bufferul de imprimanta si faptul ca permite incarcarea oricarui fel de fisier , indiferent de existenta header-ului si de valoarea octetului de flag.

Prezentam in continuare acest program , pe care l-am numit ASCII SEEKER (cautator ASCII) :

```
  10 REM xxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxx
  20 REM in linia anterioara sin
t 35 de caractere "x" dupa REM
  30 FOR a=23760 TO 23760+34
       READ x
       POKE a,x
     NEXT a
  40 DATA 33,0,88,84,93,19,1,0,1
,54,0,237,176,1,0,2,54,7,237,176
,221,33,0,64,17,48,3,175,55,205,
86,5,195,3,64
```

```
 5 ;*************************************
   ;*    THE ASCII SEEKER - VBS 1988    *
   ;*************************************
   ;
   ;
10        ORG  16384
11        PUT  25500
20 ;local system variables :
30 curadd:
          DEFW 23296
   bit.7:DEFB 0
40 ;main program sequence
   main :DI
          LD   SP,18431
          CALL ch.cons
          CALL panel
          LD   HL,23296
          LD   (curadd),HL
          CALL ndec.16
          XOR  A
          OUT  (#FE),A
          LD   (bit.7),A
          CALL status
          CALL memory
          CALL print.256
          JP   delete
   main.loop:
          CALL delay
          LD   A,#DF
          IN   A,(#FE)
          RRA
          JP   NC,one.byte.f
          RRA
          JP   NC,one.byte.b
          LD   A,#FB
          IN   A,(#FE)
          RRA
          JP   NC,one.line.b
          RRA
          JP   NC,one.page.b
          LD   A,#FD
          IN   A,(#FE)
          RRA
          JP   NC,one.line.f
          RRA
          JP   NC,one.page.f
          RRA
          JP   NC,delete
          LD   A,#7F
          IN   A,(#FE)
          RRA
          JP   NC,flip.7
          RRA
          RRA
          RRA
          RRA
          JP   NC,0
          LD   A,#BF
          IN   A,(#FE)
          RRA
          RRA
          JP   NC,load
          RRA
          RRA
          RRA
          JP   NC,home
          JP   main.loop
50 ;
   ;optional branches from main program
   ;
   one.byte.f:
          LD   HL,(curadd)
          INC  HL
   common.1:
          LD   (curadd),HL
          CALL ndec.16
          CALL memory
          CALL print.256
          JP   main.loop
   ;
   one.byte.b:
          LD   HL,(curadd)
          DEC  HL
          JP   common.1
   ;
   one.line.b:
          LD   HL,(curadd)
          LD   BC,-32
          ADD  HL,BC
          JP   common.1
   ;
   one.line.f:
          LD   HL,(curadd)
          LD   BC,32
          ADD  HL,BC
          JP   common.1
   ;
   one.page.b:
          LD   HL,(curadd)
          LD   BC,-256
          ADD  HL,BC
          JP   common.1
   ;
   one.page.f:
          LD   HL,(curadd)
          LD   BC,256
          ADD  HL,BC
          JP   common.1
   ;
   flip.7:
          LD   A,(bit.7)
          XOR  1
          LD   (bit.7),A
          CALL status
          CALL print.256
          JP   main.loop
   load :LD   DE,(curadd)
          LD   HL,0
          OR   A
          SBC  HL,DE
          LD   IX,(curadd)
          LD   D,H
          LD   E,L
          CALL load.bytes
   ;
   loaded:
          CALL print.256
          XOR  A
          OUT  (#FE),A
          DI
          JP   main.loop
   ;
60 status:
          LD   A,(bit.7)
          OR   A
          JP   Z,ignore.1
          LD   IX,message.1
   common.2:
          LD   HL,20665
          LD   (df.cc),HL
          LD   A,%1000111
          LD   (attribute),A
          LD   C,6
   newchar:
          LD   A,(IX+0)
          CALL print.1
          LD   HL,df.cc
```

```
        INC  (HL)
        INC  IX
        DEC  C
        JP   NZ,newchar
        RET
 ignore.1:
        LD   IX,message.2
        JP   common.2
 message.1:
        DEFM "RESET "
 message.2:
        DEFM "IGNORE"
70 ;
 load.bytes:
        XOR  A
        SCF
        EX   AF,AF'
        LD   A,#F
        OUT  (#FE),A
        IN   A,(#FE)
        RRA
        AND  #20
        OR   #4
        LD   C,A
        CP   A
        JP   #56B
 ;
80 memory:
        LD   IX,string
        LD   HL,20651
        LD   (df.cc),HL
        LD   HL,(curadd)
        LD   A,H
        CP   #5B
        LD   A,%1000010
        JP   C,warning
        LD   A,%1000111
 warning:
        LD   (attribute),A
        LD   C,5
 newdigit:
        LD   A,(IX+0)
        CALL print.1
        LD   HL,df.cc
        INC  (HL)
        INC  IX
        DEC  C
        JP   NZ,newdigit
        RET
100 ;
 division:
        LD   A,#30
 repeat:
        ADD  HL,BC
        INC  A
        JP   C,repeat
        SBC  HL,BC
        DEC  A
        LD   (DE),A
        INC  DE
        RET
 ;
 ndec.16:
        LD   HL,(curadd)
        LD   DE,string
        LD   BC,-10000
        CALL division
        LD   BC,-1000
        CALL division
        LD   BC,-100
        CALL division
        LD   BC,-10
        CALL division
        LD   A,L
        ADD  A,#30
        LD   (DE),A
        RET
 ;
 string:
        DEFS 5
 ;
110 df.cc:DEFW 18432
 attribute:
        DEFB #38
 ;
 print.1:
        PUSH HL
        LD   L,A
        LD   H,0
        ADD  HL,HL
        ADD  HL,HL
        ADD  HL,HL
        LD   DE,character.set-256
        ADD  HL,DE
        LD   DE,(df.cc)
        LD   B,8
 next.row:
        LD   A,(HL)
        LD   (DE),A
        INC  HL
        INC  D
        DJNZ next.row
        LD   A,D
        RRCA
        RRCA
        RRCA
        DEC  A
        AND  3
        OR   #58
        LD   D,A
        LD   A,(attribute)
        LD   (DE),A
        POP  HL
        RET
 ;
200 ;
 print.256:
        LD   HL,18432
        LD   (df.cc),HL
        LD   HL,(curadd)
 print.loop:
        LD   A,%1000110
        LD   (attribute),A
        LD   A,(HL)
        CP   129
        JP   NC,test.bit.7
        CP   32
        JP   C,invalid.char
 entry.point:
        CALL print.1
        PUSH HL
        LD   HL,df.cc
        INC  (HL)
        POP  HL
        RET  Z
        INC  HL
        JP   print.loop
 invalid.char:
        XOR  A
        LD   (attribute),A
        JP   entry.point
 test.bit.7:
        PUSH AF
        LD   A,(bit.7)
        OR   A
        JP   Z,bit.7.ignored
        LD   A,%1000101
        LD   (attribute),A
        POP  AF
        RES  7,A
        CP   32
        JP   C,invalid.char
        JP   entry.point
 bit.7.ignored:
        POP  AF
        JP   invalid.char
500 delay:LD B,200
 delay.loop.1:
        LD   A,90
 delay.loop.2:
        DEC  A
        JR   NZ,delay.loop.2
        DJNZ delay.loop.1
        RET
510 ;
 panel:LD  A,#CC
        LD   HL,20480
        EX   AF,AF'
        LD   A,#33
        LD   B,8
 panel.loop.1:
        PUSH BC
        EX   AF,AF'
        LD   B,0
 panel.loop.2:
        LD   (HL),A
        INC  HL
        DJNZ panel.loop.2
        POP  BC
        DJNZ panel.loop.1
        LD   HL,20514
        LD   (df.cc),HL
        LD   A,#42
        LD   (attribute),A
        LD   C,11
        LD   IX,message.3
        CALL newchar
        LD   HL,20525
        LD   (df.cc),HL
        LD   A,#47
        LD   (attribute),A
        LD   C,18
        LD   IX,message.4
        CALL newchar
        LD   HL,20546
        LD   (df.cc),HL
        LD   A,#42
        LD   (attribute),A
        LD   C,29
        LD   IX,message.5
        CALL newchar
        LD   HL,20642
        LD   (df.cc),HL
        LD   A,#47
        LD   (attribute),A
        LD   C,12
        LD   IX,message.6
        CALL newchar
        LD   HL,20657
        LD   (df.cc),HL
        LD   C,13
        LD   IX,message.7
        JP   newchar
 ;
```

```
    message.3:
          DEFM "Artwork by "
    message.4:
          DEFM "VIDEO BYTE STUDIOS"
    message.5:
          DEFM "   Designed September 1988   "
    message.6:
          DEFM "Memory        "
    message.7:
          DEFM "Bit 7         "
    ;
520 delete:
          LD   HL,0
          LD   DE,(curadd)
          OR   A
          SBC  HL,DE
          LD   B,H
          LD   C,L
          LD   H,D
          LD   L,E
          INC  DE
          LD   (HL),32
          LDIR
          CALL print.256
          JP   main.loop
    ;
530 ch.cons:
          LD   HL,#3D00
          LD   DE,character.set
          LD   BC,768
          LDIR
          LD   HL,character.set
          LD   BC,96
    ch.cons.loop:
          SLA  (HL)
          INC  HL
          SLA  (HL)
          INC  HL
          SLA  (HL)
          INC  HL
          INC  HL
          INC  HL
          SRA  (HL)
          INC  HL
          SRA  (HL)
          INC  HL
          SRA  (HL)
          INC  HL
          DEC  BC
          LD   A,B
          OR   C
          JR   NZ,ch.cons.loop
          RET

580 home :LD   HL,23296
          LD   (curadd),HL
          CALL ndec.16
          CALL memory
          CALL print.256
          JP   main.loop
599 ;
    end.of.file:
          NOP
    ;
600 ;
    character.set:
          DEFS 768
    ;
610 saver:LD   IX,25500
          LD   DE,816
          XOR  A
          CALL 1218
          RET
620 ;
    save :DEFL saver+25500-16384
```

Pentru instalare , introduceti intii scurtul program BASIC si rulati-l. Stergeti apoi liniile 20,30 si 40 si introduceti linia :

20 RANDOMIZE USR 23760

Introduceti ca si comanda directa CLEAR , apoi salvati programul BASIC pe banda cu :

SAVE CHR$ 22+CHR$ 1+CHR$ 0+"SEEKER" LINE 0

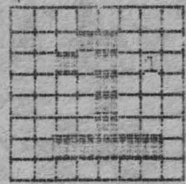
Incarcati un asamblor (va recomandam LASER GENIUS) si inserati programul in limbaj de asamblare. Salvati fisierul sursa pe o alta caseta , apoi asamblati (ATENTIE ! Asamblare cu dispunere deplasata a codului obiect !). Reveniti in BASIC , pregatiti in casetofon caseta cu "SEEKER" si porniti inregistrarea. Abia dupa aceasta introduceti RANDOMIZE USR 27085 (din GENIUS puteti face acelasi lucru direct prin EXECUTE save).

ASCII SEEKER este acum disponibil si va pune la dispozitie urmatoarele comenzi :

- Q , A - cautare linie cu linie
- W , S - cautare pagina cu pagina
- O , P - cautare caracter cu caracter
- [D]elete - sterge spatiul de stocare ASCII , de la adresa MEMORY pina la capatul RAM-ului.
- [L]oad - incarca un fisier de pe banda incepind de la adresa MEMORY.
- [H]ome - MEMORY la inceputul spatiului de stocare (23296).
- SPACE - schimba modul de lucru :
  - in modul RESET , pentru toti octetii de valoare mai mare decit 127 , se reseteaza bitul 7 si daca valoarea obtinuta este mai mare decit 31 , se tipareste caracterul ASCII corespunzator (in culoare diferita).
  - in modul IGNORE , nu se iau in considerare valorile mai mari de 127.
- [B]asic - reseteaza sistemul si revine in BASIC.

Programul descris mai sus va va ajuta sa gasiti fara probleme vocabularele multor aventuri si , nu in ultimul rind , comenzile majoritatii programelor utilitare la care nu aveti documentatie (astfel au fost depistate si comenzile si functiile G.A.C.).

Spectrum retine in ROM structura binara a celor 96 de simboluri alfanumerice , alocind cite 8 octeti fiecarui simbol. Fiecare octet contine 8 biti , astfel ca fiecare caracter poate fi privit ca o matrice de 8 x 8 biti , construita ca si in cazul exemplului din (fig.3):



```
 0 = % 00000000
24 = % 00011000
40 = % 00101000
 8 = % 00001000
 8 = % 00001000
 8 = % 00001000
62 = % 00111110
 0 = % 00000000
```

FIG 3

Pentru intregul set de caractere rezulta un total de 96 x 8 = 768 octeti. Cei 768 octeti continind setul de caractere standard din ROM sint stocati incepind de la adresa hexa 3D00.

Lucrind sub sistemul de operare BASIC , putem modifica setul de caractere , construind un nou set in RAM si modificind valoarea variabilei de sistem CHARS , definita ca si continind cu 256 mai putin decit adresa setului de caractere. Aceasta definitie pare ciudata , daca nu observam ca primul caracter din set , SPACE , are codul ASCII 32. Avem 32 x 8 = 256 , deci putem gasi foarte simplu adresa la care sint stocate datele binare ale unui caracter particular , inmultind codul caracterului cu 8 si adunindu-l la valoarea continuta in CHARS.

Rutina de tiparire din ROM poate fi apelata incarcind codul ASCII al caracterului de tiparit in acumulator (registrul A) si executind un RST #10. Aceasta rutina este insa foarte inceata si greoaie in utilizare. Odata apelata , ea trebuie sa decida printre altele daca se tipareste la consola , in partea de sus a ecranului sau la imprimanta , daca se incearca utilizarea unor coduri speciale de control , daca se tipareste un caracter normal sau un UDG.

Vom folosi asadar o rutina proprie de tiparire , mult mai rapida si potrivita scopurilor noastre. Dar inainte de a o descrie , sa mai luam in considerare un aspect : pentru a modifica setul de caractere trebuie sa renuntam benevol la 768 de octeti de memorie , iar in cazul in care vrem sa folosim mai multe seturi de caractere , cantitatea de memorie pierduta devine importanta. Nu ne permitem risipa de memorie , pentru a ne ramine un spatiu cit mai mare pentru bazele de date ale aventurii.

Rutina de tiparire pe care o vom prezenta in continuare permite utilizarea a 256 de seturi de caractere distincte , construite sintetic plecind de la setul de caractere din ROM si ocupa un spatiu de memorie mai mic decit un set de caractere obisnuit.

Parametrii de intrare ai rutinei sint :

A = codul ASCII al caracterului.
C = pozitia de tiparire dupa axa x.
B = pozitia de tiparire dupa axa y.
(font.pattern) = model binar care defineste setul de caractere
(attribute) = atributul video curent.

Modelul binar al setului de caractere contine o codificare pe 8 biti , fiecare bit specificind cite o anumita prelucrare efectuata asupra setului standard , dupa cum se arata in (fig.4):

Pot fi efectuate si combinatii pentru a obtine diferite tipuri de caractere.

Iata in continuare procesorul de tiparire , pe care l-am numit POWER PRINT , completat de o demonstratie (DEMO) :

```
Bit 0 > Right slant italics
Bit 1 > Left slant italics
Bit 2 : [illegible]
Bit 3 » [illegible]
Bit 4 > Thick
Bit 5 > Fat
Bit 6 > Fancy
Bit 7 > Curved
```

FIG.4

```
10 ;*********************************
   ;* POWER PRINT > High Speed Print *
   ;*     Video Byte Studios 1989    *
   ;*********************************
   ;
20 ;ENTRYS : A = ASCII code
   ;         B = line (y coord)
   ;         C = column (x coord)
   ; (font.pattern) = binary selector
   ; (attribute) = video colour
   ;
30 attribute:
        DEFB 0
   font.pattern:
        DEFB 0
   workspace:
        DEFS 8
   ;
40 POWER.PRINT:
        LD   L,A
        LD   H,0 ;HL=ASCII code
        ADD  HL,HL
        ADD  HL,HL
        ADD  HL,HL
        EX   DE,HL ;DE=8 x CODE
        LD   HL,#3C00 ;HL=CHARS (ROM)
        ADD  HL,DE
        EX   DE,HL ;DE=character address
   ;
50      LD   A,B
        AND  #F8
```

```
        ADD  A,#40
        LD   H,A
        LD   A,B
        AND  7
        RRCA
        RRCA
        RRCA
        ADD  A,C
        LD   L,A ;HL=display file address
        PUSH HL
        EX   DE,HL ;HL=character address
        LD   DE,workspace
        LD   BC,8
        LDIR  ;moves the char pattern in workspace
        LD   A,(font.pattern)
        SRA  A
        CALL C,right.italics
        SRA  A
        CALL C,left.italics
        SRA  A
        CALL C,rotten
        SRA  A
        CALL C,doubled
        SRA  A
        CALL C,thick
        SRA  A
        CALL C,fat
        SRA  A
        CALL C,fancy
        SRA  A
        CALL C,curved
        POP  HL ;HL=display file address
;The actual printing
;
        LD   DE,workspace ;data in workspace
        LD   B,8 ; 8 rows
next.row:
        LD   A,(DE)
        LD   (HL),A
        INC  DE
        INC  H
        DJNZ next.row
;
        LD   A,H
        RRCA
        RRCA
        RRCA
        DEC  A
        AND  3
        OR   #58
        LD   H,A ;HL=video attribute address
        LD   A,(attribute)
        LD   (HL),A
        RET
```

```
;
;Specific modification routines
;
right.italics:
        LD   HL,workspace
        SRA  (HL)
        INC  HL
        SRA  (HL)
        INC  HL
        SRA  (HL)
        INC  HL
        INC  HL
        INC  HL
        SLA  (HL)
        INC  HL
        SLA  (HL)
        INC  HL
        SLA  (HL)
        RET
;
left.italics:
        LD   HL,workspace
        SLA  (HL)
        INC  HL
        SLA  (HL)
        INC  HL
        SLA  (HL)
        INC  HL
        INC  HL
        INC  HL
        SRA  (HL)
        INC  HL
        SRA  (HL)
        INC  HL
        SRA  (HL)
        RET
;
rotten:
        PUSH AF
        LD   HL,workspace
        LD   B,4
loop.r:
        LD   A,(HL)
        AND  #AA
        LD   (HL),A
        INC  HL
        LD   A,(HL)
        AND  #55
        LD   (HL),A
        INC  HL
        DJNZ loop.r
        POP  AF
        RET
```

```
;
doubled:
        PUSH AF
        LD   HL,workspace
        LD   B,8
loop.d:
        LD   A,(HL)
        LD   C,A
        LD   A,(HL)
        SLA  A
        LD   (HL),A
        LD   A,C
        SRA  A
        OR   (HL)
        LD   (HL),A
        INC  HL
        DJNZ loop.d
        POP  AF
        RET
;
thick:LD   HL,workspace
        PUSH AF
        LD   B,8
loop.t:
        LD   A,(HL)
        SRA  A
        OR   (HL)
        LD   (HL),A
        INC  HL
        DJNZ loop.t
        POP  AF
        RET
;
fat   :PUSH AF
        LD   B,8
        LD   HL,workspace
loop.f.1:
        LD   A,(HL)
        SRA  A
        OR   (HL)
        LD   (HL),A
        INC  HL
        DJNZ loop.f.1
        LD   HL,workspace
        LD   B,7
loop.f.2:
        INC  HL
        LD   A,(HL)
        DEC  HL
        OR   (HL)
        LD   (HL),A
        INC  HL
        DJNZ loop.f.2
```

```
        POP  AF
        RET
;
fancy:LD   HL,workspace
        LD   B,4
loop.f:
        SRA  (HL)
        INC  HL
        INC  HL
        DJNZ loop.f
        RET
;
curved:
        LD   HL,workspace
        SLA  (HL)
        INC  HL
        SLA  (HL)
        INC  HL
        SLA  (HL)
        INC  HL
        INC  HL
        INC  HL
        RLC  C
        INC  HL
        SLA  (HL)
        INC  HL
        SLA  (HL)
        RET
;
;*********************************
;*      DEMONSTRATION ROUTINE    *
;*********************************
;
DEMO :LD   IX,messages
        LD   HL,16384
        LD   D,H
        LD   E,1
        LD   BC,6143
        LD   (HL),0
        LDIR
        LD   B,9
demo.loop:
        PUSH BC
        CALL demo.print
        POP  BC
        DJNZ demo.loop
        RET
;
demo.print:
        LD   A,(IX+0)
        LD   (attribute),A
        INC  IX
```

```
        LD   A,(IX+0)
        LD   (font.pattern),A
        INC  IX
        LD   B,(IX+0)
        INC  IX
        LD   C,(IX+0)
        INC  IX
dprint.loop:
        LD   A,(IX+0)
        BIT  7,A
        JR   NZ,exit
        PUSH BC
        CALL POWER.PRINT
        POP  BC
        INC  IX
        INC  C
        LD   A,C
        CP   32
        JR   NZ,dprint.loop
        LD   C,0
        INC  B
        LD   A,B
        CP   24
        RET  Z
        JR   dprint.loop
;
exit :RES   7,A
        INC  IX
        CALL POWER.PRINT
        RET
;
messages:
        DEFB #79,1,1,10,"I am itali","c"!128
        DEFB #79,2,3,8,"I am italic to","o"!128
        DEFB #7A,4,5,7,"I am really rotte","n"!128
        DEFB #3A,8,7,10,"I am double","d"!128
        DEFB #7B,16,9,11,"I am thic","k"!128
        DEFB #3B,32,11,12,"I am fa","t"!128
        DEFB #7C,64,13,11,"I am fanc","y"!128
        DEFB #3C,128,15,10,"I am curve","d"!128
        DEFB #F8,36,19,4,"And I am fat and rotte","n"!128
```

● ● ●

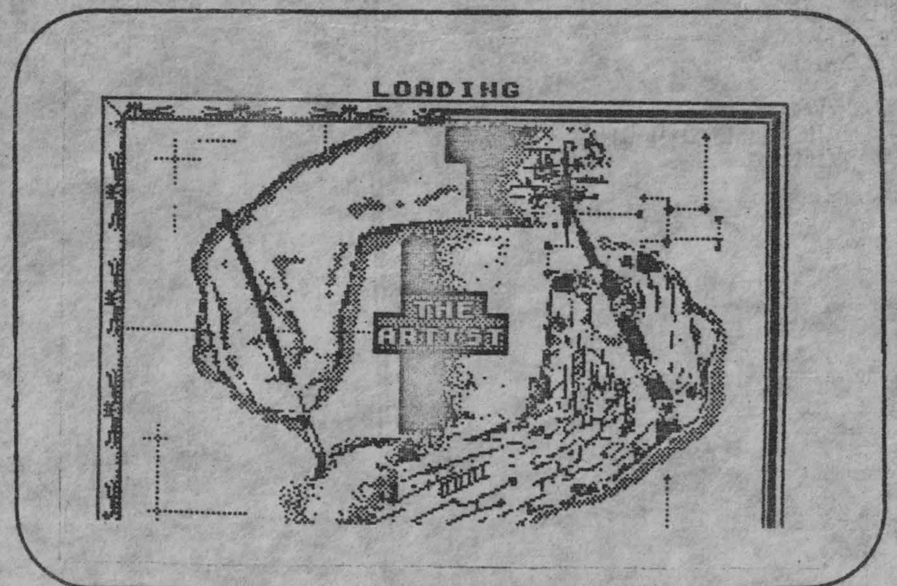
Intr-un paragraf ulterior , care se va referi la modul de realizare a editorului , vom prezenta si o rutina alternativa de tiparire (monocolor) , cu caractere avind latimea mai mica decit 8 pixeli , care permite cresterea cantitatii de informatie care poate fi afisata pe ecran la un moment dat.

Pina atunci insa , sa reluam o idee mai veche : observind ca in aventura noastra vom tipari in special litere mici , mai rar majuscule si chiar foarte rar semne speciale si cifre , ne-am gindit ca am putea mari numarul mesajelor stocate in memorie , abatindu-ne de la codul ASCII si generind un cod propriu pentru stocarea caracterelor pe 5 biti , ceva in genul anticului cod Baudot.

Aceasta tehnica de compresie este foarte utila in cazul nostru particular. Daca in schimb ar trebui sa tiparim multe majuscule sau simboluri speciale , am inregistra o pierdere de memorie , deoarece aceste caractere se retin pe 10 biti.

Daca vom dori ca la tiparire sa folosim rutina POWER PRINT va trebui sa asiguram la nivelul secventei decomprimate o conversie de la codul nostru propriu la codul ASCII (setul de caractere din ROM , utilizat de POWER PRINT la generarea automata a noilor seturi de caractere corespunde codului ASCII).

Am creat deci un cod alfanumeric nou , pe care l-am numit VBS_5 , cod complet definit de urmatorul tabel :



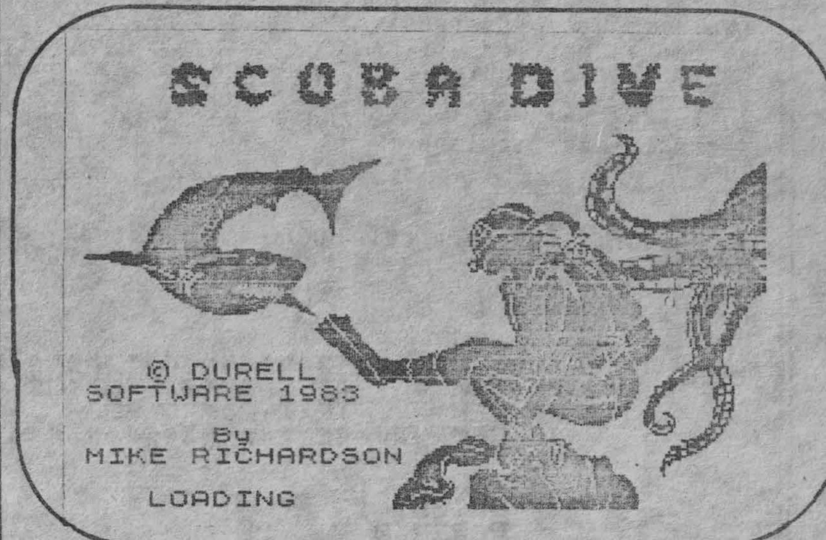
CODUL VBS_5		SECVENTA 1		SECVENTA 2		SECVENTA 3	
zecimal	binar	simbol	ASCII	simbol	ASCII	simbol	ASCII
0	00000	NEFOLOSIT		NEFOLOSIT		NEFOLOSIT	
1	00001	NEFOLOSIT		NEFOLOSIT		NEFOLOSIT	
2	00010	salt la secv 2		!	33	^	94
3	00011	salt la secv 3		"	34	`	96
4	00100	SPACE	32	#	35	?	63
5	00101	@	127	$	36	@	64
6	00110	a	97	%	37	A	65
7	00111	b	98	&	38	B	66
8	01000	c	99	'	39	C	67
9	01001	d	100	(	40	D	68
10	01010	e	101	)	41	E	69
11	01011	f	102	*	42	F	70
12	01100	g	103	+	43	G	71
13	01101	h	104	,	44	H	72
14	01110	i	105	-	45	I	73
15	01111	j	106	.	46	J	74
16	10000	k	107	/	47	K	75
17	10001	l	108	0	48	L	76
18	10010	m	109	1	49	M	77
19	10011	n	110	2	50	N	78
20	10100	o	111	3	51	O	79
21	10101	p	112	4	52	P	80
22	10110	q	113	5	53	Q	81
23	10111	r	114	6	54	R	82
24	11000	s	115	7	55	S	83
25	11001	t	116	8	56	T	84
26	11010	u	117	9	57	U	85
27	11011	v	118	:	58	V	86
28	11100	w	119	;	59	W	87
29	11101	x	120	<	60	X	88
30	11110	y	121	=	61	Y	89
31	11111	z	122	>	62	Z	90

Intrucit insa locatiile de memorie sint organizate la nivel de octet , codul nostru va trebui stocat in modul urmator :

```
  bit 7                                       bit 0     pozitia

  b22    b12    b02 ! b41    b31    b21    b11    b01    octetul (1)
  b04 ! b43    b33    b23    b13    b03 ! b42    b32    octetul (2)
  b35    b25    b15    b05 ! b44    b34    b24    b14    octetul (3)
  b17    b07 ! b46    b36    b26    b16    b06 ! b45    octetul (4)
  b48    b38    b28    b18    b08 ! b47    b37    b27    octetul (5)
```

in care bxy reprezinta bitul cu ordinul x din caracterul cu numarul y. Dupa 5 octeti (pozitii) , respectiv dupa 8 caractere ,



modelul de stocare se repeta identic.

Studiind definitia codului VBS_5 , se observa ca exista de fapt 3 coduri alternative. Trecerea de la codul de baza (1) , la oricare alta secventa se face utilizind caracterele de escape (2,respectiv 3). Tranzitia inversa se realizeaza automat. Pentru a retine de exemplu simbolul '?' , vom folosi secventa binara :

00011 ¦ 00100 ,

folosind 10 biti in loc de 5 , dupa care ne vom gasi in codul de baza.

Se mai observa ca valorile 0 (% 00000) si 1 (% 00001) nu sint folosite. Aceasta este imperios necesar , deoarece vom testa sfirsitul unei descrieri sau al unui mesaj prin intermediul unui octet de valoare 0. Fiecare din pozitiile 1,2,4 si 5 contin cite un caracter VBS_5 complet si fragmente din cel putin inca un caracter VBS_5. Intrucit caracterele VBS_5 nu pot lua valoarea 0 , putem fi siguri ca un octet nul situat pe una din aceste pozitii semnifica 'sfirsit de mesaj' si nu altceva. Singura posibilitate de ambiguitate ar fi pe pozitia 3 , unde se gasesc fragmente a doua caractere VBS_5. Din acest motiv am interzis si folosirea valorii 1 , pentru a lua masuri de precautie impotriva secventei particulare :

1 0000 ¦ 0000 1

Avind in vedere cele enuntate mai sus , am scris un program utilitar , pe care l-am denumit TEXT COMPRESSOR si care realizeaza comprimarea unor mesaje introduse de la tastatura , conform noului cod VBS_5.

```
  10 REM ########################
  20 REM #  VIDEO BYTE STUDIOS #
  30 REM #   TEXT COMPRESSOR   #
  40 REM ########################
  50 REM
  60 INK 5
     BRIGHT 1
     PAPER 0
     BORDER 0
     CLEAR 64999
     LOAD ""CODE 65000
     CLS
     PRINT AT 4,9; INVERSE 1;"TE
XT COMPRESSOR"' INVERSE 0'" the
VBS-5 code conversion kit";AT 21
,18;"VBS 1988"
  70 LET reall=0
     DIM v(3)
     POKE 65000,232
     POKE 65001,128
     REM ?current address=33000?
  80 LET mem=32000
     REM ?input buffer?
  90 PAUSE 0
     LET message=0
 100 CLS
     PRINT AT 0,0;"MESSAGE NUMBE
R         ";message'"MEMORY USED
          ";PEEK 65000+256*PEE
K 65001-33000
     GO SUB 280
 110 IF (PEEK 65000+256*PEEK 650
01-33000)>=30000 THEN PRINT AT 1
0,6; FLASH 1;"MEMORY FULL"; FLAS
H 0
 120 IF PEEK 65000+256*PEEK 6500
1<>33000 THEN PRINT AT 0,28;100-
INT (100*(PEEK 65000+256*PEEK 65
001-33000)/reall);" %"
 130 LET a$=INKEY$
     IF a$="i" OR a$="I" THEN CL
S
     PRINT AT 0,0;"MESSAGE ";m
essage;" :"''
     INPUT LINE m$
     PRINT m$
     IF LEN m$<=480 THEN GO SU
B 280
 140 IF a$="s" OR a$="S" THEN LE
T v(3)=reall
     LET v(1)=PEEK 65000+256*P
EEK 65001
     LET v(2)=message
     INPUT "Filename : ";n$
     LET n$=n$+"          "
     LET n$=n$( TO 10)
     SAVE n$ DATA v()
     SAVE n$( TO 10)CODE 33000
,PEEK 65000+256*PEEK 65001-33000
     CLS
     GO SUB 280
     GO TO 100
 150 IF a$="l" OR a$="L" THEN IN
PUT "Filename : ";n$
     PRINT AT 5,0;
     LOAD n$ DATA v()
     LOAD ""CODE
     POKE 65000,v(1)-256*INT (
v(1)/256)
     POKE 65001,INT (v(1)/256)
     LET reall=v(3)
     LET message=v(2)
     LET mem=32000
     GO TO 100
```

```
 160 IF a$="W" OR a$="w" THEN CL
S
      INPUT "MESSAGE ? ";x
      IF x<message THEN POKE 65
143,x-256*INT (x/256)
        POKE 65144,INT (x/256)
        CLS
        PRINT AT 0,0;
        RANDOMIZE USR 65167
        PAUSE 0
        GO TO 100
 170  IF a$="w" OR a$="W" THEN G
O TO 160
 180 IF a$<>"c" AND a$<>"C" THEN
 GO TO 130
 190 LET reall=reall+LEN m$
     FOR a=1 TO LEN m$
 200    IF m$(a)=" " THEN POKE me
m,4
        LET mem=mem+1
 210    IF m$(a)=" THEN POKE me
m,5
        LET mem=mem+1
 220    IF CODE m$(a)>=97 AND COD
E m$(a)<=122 THEN POKE mem,(CODE
 m$(a)-91)
        LET mem=mem+1
 230    IF CODE m$(a)>=33 AND COD
E m$(a)<=62 THEN POKE mem,2
        POKE mem+1,(CODE m$(a)-
31)
        LET mem=mem+2
 240    IF m$(a)="^" THEN POKE me
m,3
        POKE mem+1,2
        LET mem=mem+2
 250    IF m$(a)="`" THEN POKE me
m,3
        POKE mem+1,3
        LET mem=mem+2
 260    IF CODE m$(a)>=63 AND COD
E m$(a)<=90 THEN POKE mem,3
        POKE mem+1,(CODE m$(a)-
59)
        LET mem=mem+2
 270 NEXT a
     POKE mem,0
     LET mem=mem+1
     RANDOMIZE USR 65002
     LET message=message+1
     LET mem=32000
     GO TO 100
 280 REM ?HELP LINE?
 290 PRINT AT 17,0;"I > Input cu
rrent message"'"C > Compress mes
sage"'"W > Write desired message
"'"S > Save database"'"L > Load
database"
 300 RETURN
```

```
 1 *LLIST   ON
10 ;
   ; ** SUPER ADVENTURE SYSTEM **
   ; (c) VIDEO BYTE SYSTEM (SOFTWARE)
   ;     dec. 1988 to xxx. 1989
   ;
   ; by DRAGHI  & TEEB
   ;
   ; 29 dec. 1988 h:00:01:00 >working hard
   ;
   ; TEXT COMPRESSING ROUTINES FOR GENERAL
   ;                PURPOUSES
   ;
20      ORG  65000
        PUT  25500
   ;
30 mem.pointer:
        DEFW comp.Dbase
   ;
40 text.compressor:
        LD   HL,input.buff
        LD   DE,(mem.pointer)
   ;
   comp.loop:
        LD   A,(HL)
        OR   A ; end of buffer entrys
        JR   Z,exit.1 ; marked by a zero
        AND  #1F ; % 00011111
        LD   B,A
        INC  HL
        LD   A,(HL)
        OR   A
        JR   Z,exit.2
        AND  #7 ; % 00000111
        RRCA
        RRCA
        RRCA  ; positioning
        OR   B
        LD   (DE),A
        INC  DE ; next DBase address
        LD   A,(HL)
        AND  #18 ; % 00011000
        RRCA
        RRCA
        RRCA
        LD   B,A
        INC  HL
        LD   A,(HL)
        OR   A
        JR   Z,exit.2
        AND  #1F ; % 00011111
        RLCA
        RLCA
        OR   B
        LD   B,A
        INC  HL
        LD   A,(HL)
        OR   A
        JR   Z,exit.2
        AND  #1 ; % 00000001
        RRCA
        OR   B
        LD   (DE),A
        INC  DE
        LD   A,(HL)
        AND  #1E ; % 00011110
        RRCA
        LD   B,A
        INC  HL
        LD   A,(HL)
        OR   A
        JR   Z,exit.2
        AND  #F ; % 00001111
        RRCA
        RRCA
        RRCA
        RRCA
        OR   B
        LD   (DE),A
        INC  DE
        LD   A,(HL)
        AND  #10 ; % 00010000
        RRCA
        RRCA
        RRCA
        RRCA
        LD   B,A
        INC  HL
        LD   A,(HL)
        OR   A
        JR   Z,exit.2
        AND  #1F ; % 00011111
        RLCA
        OR   B
        LD   B,A
        INC  HL
        LD   A,(HL)
        OR   A
```

```
        JR   Z,exit.2
        AND  #3 ; % 00000011
        RRCA
        RRCA
        OR   B
        LD   (DE),A
        INC  DE
        LD   A,(HL)
        AND  #1C ; % 00011100
        RRCA
        RRCA
        LD   B,A
        INC  HL
        LD   A,(HL)
        OR   A
        JR   Z,exit.2
        AND  #1F ; % 00011111
        RLCA
        RLCA
        RLCA
        OR   B
        LD   (DE),A
        INC  DE
        INC  HL
        JR   comp.loop
;
;
exit.2:
        LD   A,B
        LD   (DE),A
        OR   A
        JR   Z,exit.3
        INC  DE
exit.1:
        XOR  A
        LD   (DE),A
exit.3:
        INC  DE
        LD   (mem.pointer),DE
        RET
;
;
comp.Dbase:
        EQU  33000 ; assuring 1000 bytes for the input buffe

input.buff:
        EQU  32000
;
;
;
;
;
mess.no:
        DEFW 1 ;message number...Now you see the use of CARG
```



```
;
;
;
point.to.message:
        LD   HL,(mess.no) ;pretty name
        EX   DE,HL
        LD   HL,comp.Dbase
        LD   A,D
        OR   E
        RET  Z ; return if message zero
scan.loop:
        INC  HL
        LD   A,(HL)
        OR   A ;one 'end.of.message'
        JR   NZ,scan.loop ; is pointed by a zero
        DEC  DE
        LD   A,D
        OR   E
        JR   NZ,scan.loop ; scan (mess.no) messages
        INC  HL ; however skip the ending zero
        RET
;
;
MESSAGE:
;push hl
;push de
;push bc
;push af
        CALL point.to.message
        CALL txt.decomp
;pop af
;pop bc
;pop de
;pop hl
;
;^^^^ Eh ....  ????? ^^^^
;
        RET
;
;
txt.decomp:
        LD   A,(HL)
        OR   A
        RET  Z
        AND  #1F ; % 00011111
        CALL ASCII.converter ; convert & print
        LD   A,(HL)
        AND  #E0 ; % 11100000
        RLCA
        RLCA
        RLCA
        LD   B,A
        INC  HL
        LD   A,(HL)
```

```
        OR   A
        JR   Z,ret.point
        AND  #3 ; % 00000011
        RLCA
        RLCA
        RLCA
        OR   B
        CALL ASCII.converter
        LD   A,(HL)
        AND  #7C ; % 01111100
        RRCA
        RRCA
        CALL ASCII.converter
        LD   A,(HL)
        AND  #80 ; % 10000000
        RLCA
        LD   B,A
        INC  HL
        LD   A,(HL)
        OR   A
        JR   Z,ret.point
        AND  #F ; % 00001111
        RLCA
        OR   B
        CALL ASCII.converter
        LD   A,(HL)
        AND  #F0 ; % 11110000
        RRCA
        RRCA
        RRCA
        RRCA
        LD   B,A
        INC  HL
        LD   A,(HL)
        OR   A
        JR   Z,ret.point
        AND  #1 ; % 00000001
        RLCA
        RLCA
        RLCA
        RLCA
        OR   B
        CALL ASCII.converter
        LD   A,(HL)
        AND  #3E ; % 00111110
        RRCA
        CALL ASCII.converter
        LD   A,(HL)
        AND  #C0 ; % 11000000
        RLCA
        RLCA
        LD   B,A
        INC  HL
```

```
        LD   A,(HL)
        OR   A
        JR   Z,ret.point
        AND  #7 ; % 00000111
        RLCA
        RLCA
        OR   B
        CALL ASCII.converter
        LD   A,(HL)
        AND  #F8 ; % 11111000
        RRCA
        RRCA
        RRCA
        CALL ASCII.converter
        INC  HL
        JR   txt.decomp
 85 ;
    ret.point:
        LD   A,B ; try to print the partial character
        CALL ASCII.converter
        RET    ; and then return
    ;
 90 ;
    ;
    shift:DEFB 0 ;shift marker = 0 for set.1,2 for set.2 and 3
for set.3
    ;
    ASCII.converter:
        OR   A
        RET  Z ;ignore the illegal characters
        CP   1
        RET  Z
        LD   B,A ;hold the VBS-5 code
        LD   A,(shift)
        CP   2
        JR   Z,set.2
        CP   3
        JR   Z,set.3 ; select the chars.set
        LD   A,B ; recall the VBS-5 code
        CP   4 ; is it less then four ?
        JR   NC,next.1
        LD   (shift),A ; if yes then it is a shift
        RET    ; and nothing else
    next.1:
        JR   NZ,next.2 ;if it isn't four,skip forward
        LD   A,32 ; but if it is four then is a SPACE
        CALL print ; so print it
        RET
    next.2:
        CP   5
        JR   NZ,next.3
        LD   A,127 ; the (c) symbol
        CALL print
        RET
    next.3:
        LD   B,91 ; displacement till small letters
        ADD  A,B
        CALL print
        RET
    ;
    set.2:LD A,B ; recall code
        LD   B,31 ; displacement till symbols
        ADD  A,B
        CALL print
    clear.shift:
        XOR  A
        LD   (shift),A
        RET
    ;
    set.3:LD A,B
        CP   2
        JR   NZ,next.4
        LD   A,94 ; the ^ symbol
        CALL print
        JR   clear.shift
    next.4:
        CP   3
        JR   NZ,next.5
        LD   A,96 ; the ' symbol
        CALL print
        JR   clear.shift
    next.5:
        LD   B,59 ; displacement till capitals
        ADD  A,B
        CALL print
        JR   clear.shift
100 ;
    ;
    print:PUSH HL
        RST  16
        POP  HL
        RET    ; and it is only for test
110 ;
    end.of.file:
```

In urma folosirii acestui utilitar se obtine un fisier text comprimat (fisierul Bytes) , pe care il vom putea utiliza in a-ventura noastra (sau in alte programe care solicita mult text , cum ar fi de exemplu editoarele de tesxt) , dar trebuie sa avem grija sa adaugam si rutina de decomprimare si conversie ASCII (liniile 70..100 - in linia 100 se poate apela bineinteles si POWER PRINT in loc de RST #10). Tabloul numeric ajuta la conser-varea starii programului , atunci cind vrem sa oprim lucrul si sa continuam editarea aceluiasi fisier cu alta ocazie.

Pentru instalarea programului se introduce intii sectiunea BASIC care se salveaza pe caseta cu :

```
SAVE CHR$ 22+ CHR$ 1+CHR$ 0+"TXTCOM" LINE 0
```

apoi se incarca un asamblor si se insereaza programul in limbaj de asamblare. Se asambleaza si se salveaza pe caseta codul obiect , cu :

```
SAVE CHR$ 22+CHR$ 3+CHR$ 0+"TXTOBJ" CODE 25500,392
```

Procentajul din coltul din dreapta sus caracterizeaza eficienta compresiei.

Acestea fiind spuse , consideram ca am rezolvat in mare masura problema comunicarii CALCULATOR -> UTILIZATOR prin intermediul ecranului si ne vom referi in continuare la comunicarea inversa UTILIZATOR -> CALCULATOR , care se face prin intermediul tastaturii.

Tastatura calculatorului SPECTRUM are 40 de taste. Acestea par a fi aranjate in 4 rinduri a cite 10 taste , dar de fapt , calculatorul le vede ca fiind 8 semirinduri a cite 5 taste. Tastatura este legata de circuit prin doua legaturi cu film carbonic. Una dintre ele are 8 linii , iar cealalta 5. Cele 8 linii ale primei legaturi sint conectate la bitii 8..15 ai magistralei de adrese , in timp ce cele 5 linii ale legaturii mai mici sint conectate la bitii 0..4 ai magistralei de adrese.

Atunci cind este selectata tastatura ca si periferic , pentru decodificare calculatorul realizeaza (intr-o descriere naiva) urmatorul test (testele se realizeaza pentru fiecare semirind in parte) :

- la linia de adresa in cauza aplica un "curent" ;
- fiecare din cele 5 taste din semirindul testat poate fi considerata ca fiind un intrerupator conectat intre una din cele 5 linii de date si linia de adrese corespunzatoare semirindului, permitind la apasare trecerea "curentului" ;
- calculatorul citeste cele 5 linii de date si daca una dintre ele este strabatuta de "curent" , el stie ca tasta aferenta a fost apasata.

Putem eticheta prin conventie liniile de adrese cu A8-A15 iar liniile de date cu D0-D4. Liniile de adrese sint alocate semirindurilor dupa cum se vede in (fig.5):

De fiecare data cind vrem sa citim un semirind , punem linia sa de adrese pe nivel scazut (zero). Similar , cind o tasta a unui semirind este apasata , linia sa de date trece pe nivel scazut , in caz contrar fiind pe nivel ridicat (unu).

In cadrul unui semirind , liniile de date sint atasate tastelor ca in (fig.6):

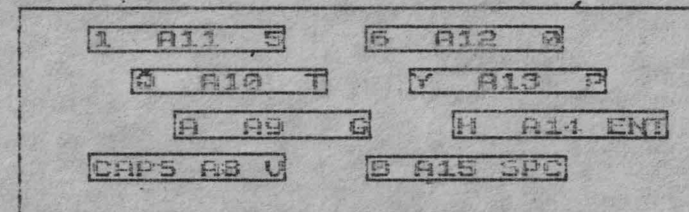


FIG.5

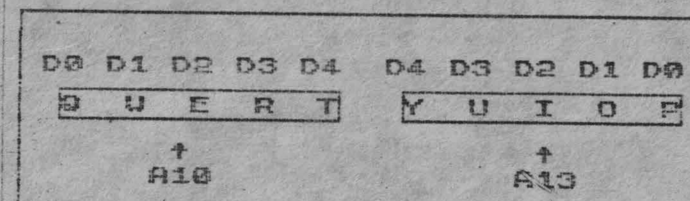


FIG.6

Tastatura insasi este selectata ca si periferic punind linia de adrese A0 pe nivel scazut. Rezulta ca octetul cel mai putin semnificativ al adresei portului de intrare pentru tastatura este #FE si la selectarea tastaturii ca si periferic vom folosi instructiunea

```
IN A,(#FE)
```

Initial A va trebui sa contina octetul cel mai semnificativ al adresei portului de intrare asociat semirindului pe care il testam (A8-A15). De exemplu , pentru a citi semirindul A-G , caruia ii corespunde linia de adrese A9 , vom utiliza secventa :

```
LD A,#FD
IN A,(#FE)
```

Pentru usurinta , prezentam in continuare o tabela cu valorile octetului cel mai semnificativ pentru fiecare semirind :

SEMIRINDUL	LINIA	BIT	OCTETUL C.M.S.
CAPS - V	A8	0	# FE = % 11111110
A - G	A9	1	# FD = % 11111101
Q - T	A10	2	# FB = % 11111011
1 - 5	A11	3	# F7 = % 11110111
6 - 0	A12	4	# EF = % 11101111
Y - P	A13	5	# DF = % 11011111
H - ENTER	A14	6	# BF = % 10111111
B - SPACE	A15	7	# 7F = % 01111111

Putem scrie acum un exemplu de program care testeaza tasta SPACE :

```
LD   A,#7f
IN   A,(#FE)
RRA               ; D0 in indicatorul CY
JP   NC,SPACE     ; Daca D0=0 -> s-a apasat SPACE
```

Este posibil sa citim chiar mai mult de un semirind la o trecere , resetind mai multi biti ai octetului cel mai semnificativ al magistralei de adrese. Pentru a testa intregul rind de jos , folosim secventa :

```
LD   A,% 01111110
IN   A,(#FE)
AND  % 00011111
XOR  % 00011111
JR   NZ,APASAT
```

Iata si un exemplu pentru secventa care urmeaza mesajului : 'Apasati orice tasta pentru a continua' :

```
XOR   A           ;toate semirindurile
```



```
PAUZA:IN    A,(#FE)     ;citim intreaga tastatura
      CPL               ;setam bitii tastelor apasate
      AND   % 00011111  ;pastram doar D0-D5 complementat
      JP    NZ,START
      JR    PAUZA
```

Acum dispunem de cunostintele necesare pentru a aborda o portiune fundamentala a sistemului nostru de operare :

E D I T O R U L

Este vorba despre un editor orientat spre programele de tip aventura. Particularitatile sale sint urmatoarele:

- este un editor pe o singura linie.
- permite scrierea cu orice marime de caractere.
- pozitionarea caracterelor se face la nivel de pixel.
- admite ca si coduri de control doar DEL(ete) si CR.
- prezinta functia de AUTOREPEAT in stilul SINCLAIR.
- scroll automat la depasirea inferioara a ecranului.
- dezvoltare foarte simpla spre orice alt tip de editor.

Desi listingul programului in limbaj de asamblare este comentat pe larg ne permitem sa revenim cu explicatii in citeva puncte considerate esentiale.

Bucla principala a editorului (linia 20) are intr-o descriere comuna urmatoarea structura:

- tipareste cursorul pentru primul caracter.
- sterge buferul (buff) (subrutina clear.buffer).
- baleeaza tastatura (subrutina get.key.code).
- converteste rezultatul baleerii intr-un cod ASCII tiparibil sau un cod de control (subrutina code.to.ASCII).

```
10 ; ******************************
   ; *** THE VBS ONE LINE EDITOR ***
   ; ******************************
20 editor:
   ;
        PUSH BC
        PUSH DE
        PUSH HL
   ;print an initial cursor
        XOR  A
        LD   (x.car),A ;tab 0
        LD   A,CURCH
        CALL print.char
   ;enter the editor
        LD   B,BULEN ;counter
        LD   HL,buff ;pointer
        CALL clear.buffer
   ed1  :LD  DE,REDEL
   ed2  :CALL get.key.code
        CALL code.to.ASCII
        CP   CR ;end with CR
        JR   Z,ed.end
        CP   DEL
        JR   Z,delete
        CP   32
        JR   C,ed1 ;control codes
        CP   128
        JR   NC,ed1 ;not ASCII
   ; the code is a valid ASCII char
        LD   (HL),A ;print in buffer
        CALL click
        CALL print.in.screen
        INC  HL
        CALL depress.key
        LD   A,D
        OR   E
        JR   Z,autorepeat
        DJNZ ed1
   ed.end:
        POP  HL
        POP  DE
        POP  BC
        RET
   ;
   autorepeat:
        LD   DE,REPER
        DJNZ ed2
        JR   ed.end
   ;
   delete:
        LD   A,B ;do nothing if
        CP   BULEN ;start of buffer
        JR   Z,del1
        DEC  HL ;back 1 position
        INC  B ;correct counter
        LD   A," " ;space
        LD   (HL),A ;delete caracter
   del1 :DEC HL ;previous char
        INC  B ;in buffer
        CALL click
        CALL print.in.screen
        INC  HL ;restore pointer
        DEC  B ;and counter
        CALL depress.key
        LD   A,D
        OR   E
        JR   NZ,ed1
        INC  B
        JR   autorepeat
   ;
30 clear.buffer:
   ;
        PUSH BC
        PUSH HL
        LD   B,BULEN
        LD   A," "
   clb1 :LD  (HL),A
        INC  HL
        DJNZ clb1
        POP  HL
        POP  BC
        RET
   ;
40 print.char:
   ;
        PUSH BC
        PUSH DE
        PUSH HL
```

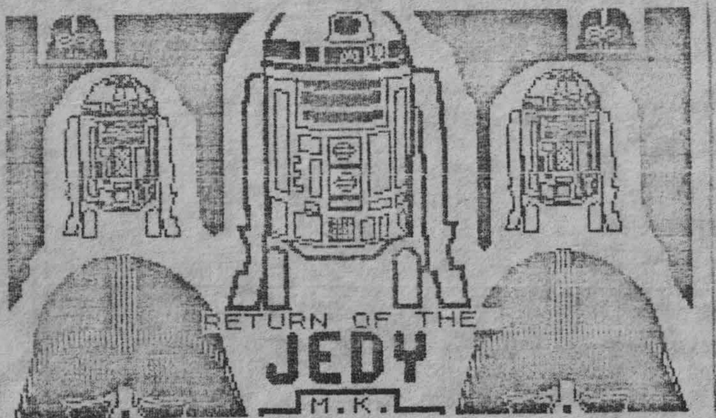
```
        EXX
        PUSH BC
        PUSH DE
        PUSH HL
        LD   HL,char.set ;base of char.set
        SUB  32 ;begins with space
        LD   E,A
        LD   D,0
        LD   B,3 ;multiply by 8
pch1 :SLA  E
        RL   D
        DJNZ pch1
        ADD  HL,DE
;hl' pointer to char pattern
        EXX
        LD   C,DYCAR
        LD   A,(y.car)
pch3 :CP   192 ;is y)=192 ?
        CALL NC,scroll.one.row ; yes
        LD   E,A ; y
        LD   A,(x.car)
        LD   D,A ;x
        CALL adr.1
;hl adress in d.f.
;b target bit
;clear area to print
        PUSH BC
        LD   DE,255
        LD   A,B
        OR   A
        JR   Z,pch6
        SCF
pch5 :RR   D
        RR   E
        DJNZ pch5
pch6 :LD   A,D
        AND  (HL)
        LD   (HL),A
        INC  L
        LD   A,E
        AND  (HL)
        LD   (HL),A
        DEC  L
        POP  BC
;the actual printing
        EXX
        LD   A,(HL) ;get pattern
        INC  HL ;next pattern byte
        EXX
        LD   D,A
        LD   E,0
        LD   A,B ;if b=0
        OR   A ;no need for
```

```
        JR   Z,pch4 ;rotation
pch2 :SRL  D
        RR   E
        DJNZ pch2
pch4 :LD   A,D
        OR   (HL)
        LD   (HL),A ;print left
        INC  L
        LD   A,E
        OR   (HL)
        LD   (HL),A ;print right
        LD   A,(y.car)
        INC  A ;next row
        LD   (y.car),A
        DEC  C ;last row ?
        JR   NZ,pch3
        SUB  DYCAR ;restore y
        LD   (y.car),A
        EXX
        POP  HL
        POP  DE
        POP  BC
        EXX
        POP  HL
        POP  DE
        POP  BC
        RET
;
50 adr.1:
;
        PUSH AF
        LD   L,D
        SRL  L
        SRL  L
        SRL  L
        LD   A,E
        SLA  A
        SLA  A
        AND  %11100000
        OR   L
        LD   L,A
        LD   A,E
        SRL  A
        SRL  A
        SRL  A
        AND  %11000
        OR   %1000000
        LD   H,A
        LD   A,E
        AND  %111
        OR   H
        LD   H,A
        LD   A,D
```

```
        AND  %111
        LD   B,A
        POP  AF
        RET
;
60 scroll.one.row:
;
        PUSH HL
        LD   HL,64 ;1'st row 2'nd 2/3
        LD   (yscrl),HL
        LD   HL,#207F ;32*127
        LD   (dyscr),HL
        CALL scroll.up
        POP  HL
        LD   A,(y.car)
        DEC  A
        LD   (y.car),A ;correct coordinates
        RET
;
scroll.up:
        PUSH BC
        PUSH DE
        PUSH HL
        LD   DE,(yscrl)
        LD   A,(dyscr)
        LD   B,A ;rows to scroll
scrl1 PUSH BC
        CALL adr.1
        PUSH HL
        INC  E
        CALL adr.1
        POP  DE
        LD   A,(dxscr)
        LD   C,A
        LD   B,0
        LDIR
        LD   DE,(yscrl)
        INC  E ;y=y+1
        LD   (yscrl),DE
        POP  BC
        DJNZ scrl1
        POP  HL
        POP  DE
        POP  BC
        RET
;
70 get.key.code:
;
;fills kcode with keybord status
        PUSH BC
        PUSH DE
        PUSH HL
        LD   B,%11111110 ;first row
```

```
        LD   C,0
        LD   E,8 ;counter
        LD   HL,kcode
gk1   :IN    A,(C)
        AND  %11111
        LD   (HL),A
        INC  HL
        DEC  E
        JR   Z,gk.end ;last row
        RLC  B ;next row
        JR   gk1
gk.end:
        POP  HL
        POP  DE
        POP  BC
        RET
 ;
80 wait.a.key:
 ;
        LD   A,%11111110
        IN   A,(254)
        AND  %11110 ;ignore CS alone
        CP   %11110
        RET  NZ
        LD   A,%1111111
        IN   A,(254)
        AND  %11101 ;ignore SS alone
        CP   %11101
        RET  NZ
        LD   A,%10000001 ;the remaining 6 rows
        IN   A,(254)
        AND  %11111
        XOR  %11111
        JR   Z,wait.a.key
        RET
 ;
depress.key:
        PUSH BC
dep1  :LD    A,%10000001
        IN   A,(254)
        AND  %11111
        XOR  %11111
        LD   C,A
        LD   A,%11111110
        IN   A,(254)
        AND  %11110
        XOR  %11110
        OR   C
        LD   C,A
        LD   A,%1111111
        IN   A,(254)
        AND  %11101
        XOR  %11101
        OR   C
        LD   C,A
        DEC  DE
        LD   A,D
        OR   E
        JR   Z,dep.ret
        LD   A,C
        OR   A
        JR   NZ,dep1
dep.ret:
        POP  BC
        RET  ;returns only if all keys are depressed ignori
ng CS and SS or time-out
 ;
90 click:
 ;
 ;keyboard click
        PUSH AF
        PUSH BC
        LD   B,CLCK
        LD   A,0
        OUT  (254),A
        LD   A,255
        OUT  (254),A
        DJNZ .
        LD   A,0
        OUT  (254),A
        POP  BC
        POP  AF
        RET
 ;
100 code.to.ASCII:
 ;
        PUSH BC
        PUSH DE
        PUSH HL
 ;
        LD   A,(kcode)
        BIT  0,A
        JR   Z,plus.caps ;caps pressed
        LD   A,(kcode+7)
        BIT  1,A
        JR   Z,plus.symb ;symb pressed
 ;simple keys
        LD   B,8
        LD   HL,kcode
cta1  :LD    A,(HL)
        CP   %11111
        JR   NZ,the.row
        INC  HL
        DJNZ cta1
        XOR  A ;very unlikely
cta.ret:
        POP  HL
        POP  DE
        POP  BC
        RET
 ;
the.row:
        LD   HL,ktable ;base of table
tr1   :LD    DE,5 ;5 keys/row
        LD   C,A ;save key code
        LD   A,8
        SUB  B
        JR   Z,pointer.ok
        LD   B,A
cta2  :ADD   HL,DE
        DJNZ cta2
pointer.ok:
        LD   B,5
        LD   A,C ;restore key code
        LD   E,%11110 ;for comparing
cta3  :CP    E
        JR   Z,the.key
        INC  HL
        RL   E
        SET  0,E
        RES  5,E
        DJNZ cta3
        XOR  A ;2 or more keys/row
        JR   cta.ret
the.key:
        LD   A,(HL) ;the ASCII code
        JR   cta.ret
 ;
plus.caps:
 ;
        LD   HL,kcode
        SET  0,(HL) ;clear CS pressed
        LD   HL,kcode+7
        LD   B,8
cta4  :LD    A,(HL)
        CP   %11111
        JR   NZ,the.caps
        DEC  HL
        DJNZ cta4
        XOR  A
        JR   cta.ret
 ;
the.caps:
        LD   HL,cstable
        JR   tr1
plus.symb:
 ;
```

```
        LD   HL,kcode+7
        SET  1,(HL) ;clear SS pressed
        LD   B,8
cta5 :LD   A,(HL)
        CP   %11111
        JR   NZ,the.sym
        DEC  HL
        DJNZ cta5
        XOR  A
        JR   cta.ret
;
the.sym:
        LD   HL,sstable
        JR   tr1
;
110 ;TABLES DECODING THE KEYBOARD
;
ktable:
        DB   0,"zxcvasdfgqwert1234509876poiuy",CR,"lkjh ",0,
"mnb"
cstable:
        DB   32,0,"MNB",CR,"LKJHPOIUY",DEL,0,0,0,0,0,0,0,0,0
,"QWERTASDFG",0,"ZXCV"
sstable:
        DB   32,0,".,#",0,"=+-^",34,59,0,93,91,95,")('&!@#$%
",0,0,0,"<>",126,124,92,123,125,0,":?/"
120 print.buffer:
;
;prints the whole buffer in the edit area
        PUSH BC
        PUSH HL
        XOR  A
        LD   (x.car),A ;1'st column
        LD   HL,buff
        LD   B,BULEN
prbf :LD   A,(HL)
        CALL print.char
        INC  HL ;next in buffer
        LD   A,(x.car) ;adjust
        ADD  A,DXCAR ;character
        LD   (x.car),A ;position
        DJNZ prbf
        POP  HL
        POP  BC
        RET
;
print.in.screen:
;
;prints the last char in buffer followed by the cursor and
 a space
        PUSH BC
        LD   A,BULEN
        CP   B ;if the 1'st char in
        JR   C,pis2 ;buff was deleted
        SUB  B
        LD   B,DXCAR ;multiply by
        DEC  B ;DXCAR to obtain the
        LD   C,A ;pixel position
pis1 :ADD  A,C
        DJNZ pis1
        LD   (x.car),A ;tab x
        LD   A,(HL) ;get current char
        CALL print.char
        LD   A,(x.car)
        ADD  A,DXCAR
pis3 :LD   (x.car),A ;tab x+1
;print the cursor and a space
        LD   A,CURCH ;get the cursor
        CALL print.char
        LD   A,(x.car)
        ADD  A,DXCAR
        LD   (x.car),A ;tab x+2
        LD   A," " ;space
        CALL print.char
        POP  BC
        RET
;
pis2 :XOR  A
        JR   pis3
;
130 ; SYMBOLS DEFINITION
;
DXCAR:EQU  7 ;width of char
DYCAR:EQU  8 ;hight of char
BULEN:EQU  32 ;lenghth of inp. buf.
CR   :EQU  13 ;caridge return code
DEL  :EQU  10 ;delete code
CURCH:EQU  95 ;cursor character
REDEL:EQU  10240 ;delay before autorepeat
REPER:EQU  512 ;autorepeat period
CLCK :EQU  #80 ;lenght of keyboard click
;
x.car:DB   0 ;character x (pix)
y.car:DB   0 ;character y
kcode:DW   0,0,0,0 ;key(s) pressed
yscrl:DB   0 ;"natural" coordinates
xscrl:DB   0 ;for the
dyscr:DB   0 ;scroll
dxscr:DB   0 ;routine
buff :DS   BULEN
;
140 ;CHARACTER SET
;
char.set:
       EQU  *23606+256
;
```



- separa codurile de control de caracterele tiparibile si de codurile nepermise (Ø).
- actioneaza adecvat conform ramificarii de mai sus.
- asteapta depresarea tuturor tastelor cu exceptia celor de CAPS SHIFT si SYMBOL SHIFT pe o perioada stabilita de REDEL (interval dupa care tasta apasata initiaza functia de AUTOREPEAT).
- repeta ultimele cinci aliniate pina la umplerea bufferului.

Stergerea bufferului consta in umplerea lui cu caracterul SPACE (ASCII 32 sau #2Ø) (linia 3Ø).

In baleerea tastaturii imaginea celor 5 biti ce decodifica tastele apasate pe fiecare dintre cele 8 rinduri este retinuta in variabilele kcode .. kcode+7 (linia 13Ø).

Pentru convertirea in cod ASCII (inclusiv coduri de control si coduri nepermise) se folosesc tabelele:

ktable - pentru taste simple
cstable - pentru taste apasate impreuna cu CAPS SHIFT
sstable - pentru taste apasate impreuna cu SYMBOL SHIFT

Ordinea in care a fosta facuta decodificarea a fost schimbata la ultimele doua tabele pentru a evita un bug necontrolat (sau o slabiciune a HARD-ului de SPECTRUM). De asemenea este cunoscuta (si inca neexplicata) comportarea caracterului (") impreuna cu functia de AUTOREPEAT.

Mai notati ca la apasarea a doua taste simultan se obtine codul tastei care a fost convertit primul. O solutie ar fi includerea in bucla principala a editorului a subrutinei wait.a.key (prezenta in listing) dar s-a observat ca nu e necesara.

Alaturi de variabila REDEL, a fost prevazut si variabila REPER care da o temporizare intre doua repetari succesive dupa initializarea procesului de AUTOREPEAT. Evident, aceste variabile pot fi modificate cu acelasi efect ca la omonimele lor din variabilele sistemului de operare BASIC.

Tiparirea unui caracter sau stergerea lui e insotita de o confirmare sonora creata de subrutina click. Variabila CLICK actioneaza asupra perioadei sunetului emis.

Pentru acest editor expresia "actioneaza adecvat" are doar trei semnificatii:

- DACA caracterul obtinut este tiparibil, el va fi inscris in buffer si tiparit pe ecran (subrutina print.in.screen). Pointerul din buffer va avansa si contorul lungime ramase disponibile in buffer se va decrementa.
- DACA caracterul obtinut este DEL (prin conventia noastra ASCII 1Ø sau #ØA) se inlocuieste ultimul caracter din buffer cu SPACE (ASCII 32 sau #2Ø), caracterul se sterge de pe ecran (tot cu print.in.screen) iar pointerul in buffer este decrementat si contorul ce da spatiul disponibil incrementat.
- DACA caracterul obtinut este CR (ASCII 13 sau #ØD) sau

contorul spatiului disponibil a ajuns la zero se iese din editor cu bufferul pregatit pentru prelucrari ulterioare.

In acest moment ar trebui sa fie clar ca asignarea unor coduri de control si altor combinatii de taste (de ex. cursoarele, TRUE VIDEO, EDIT sau SS+i,SS+u etc.) si tratarea lor adecvata conduce la realizarea altor tipuri de editoare de exemplu pentru tratarea textelor, editoare de programe pe intreg ecranul etc.

Nu vom insista prea mult asupra subrutinei print,char decit prin a spune ca stie sa tipareasca la coordonatele x.car si y.car un caracter de marimea DXCAR pe DYCAR a carui cod ASCII este transmis in acumulator. Este o subrutina foarte puternica, si o puteti folosi in multe alte aplicatii.

In listig, la linia 140 se poate defini setul de caractere ca un model binar de pixeli. Caracterele mai mici de 8 pixeli latime trebuie aliniate la stinga. In listing se arata cum poate fi folosit setul de caractere din ROM dar cu o latime de doar 7 pixeli.

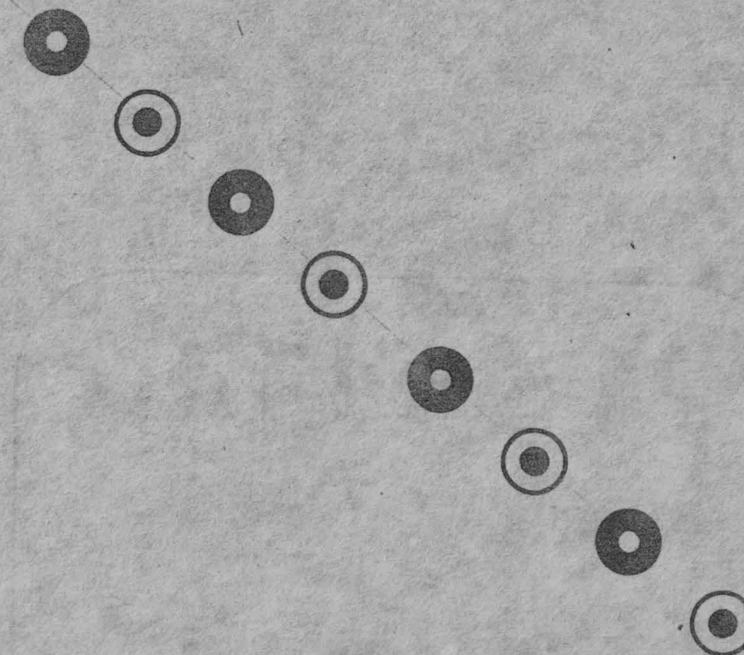
La linia 120, subrutina print.buffer va fi folosita pentru dezvoltarea ulterioara a sistemului de operare SAS (SUPER ADVENTURE SYSTEM). De asemenea, subrutina scroll.up impreuna cu variabilele xscrl,yscrl,dxscr si dyscr are o utilitate mult mai generala decit simpla apelare din scroll.one.row (linia 60).

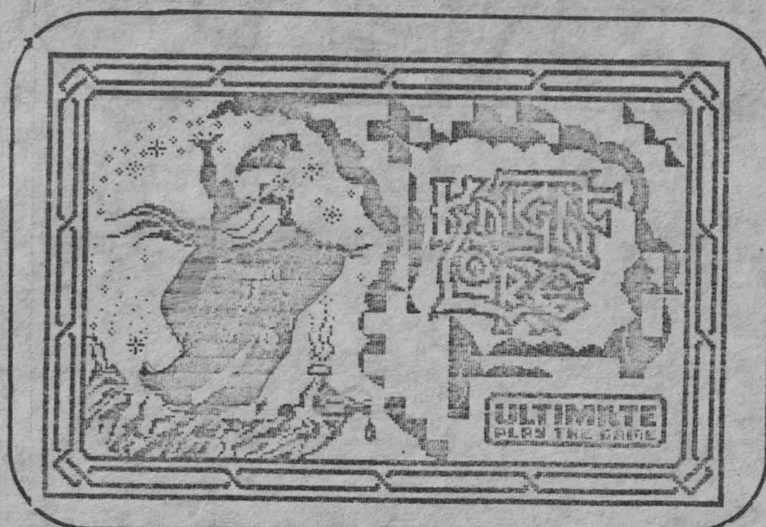
Speram ca aceasta descriere tehnica a editorului nu a reusit sa va plictiseasca. In numarul urmator vom vorbi, printre altele, de ce trebuie facut cu bufferul astfel obtinut si vom propune o prima varianta de analizor sintactic. Dar pina atunci sa continuam cu:

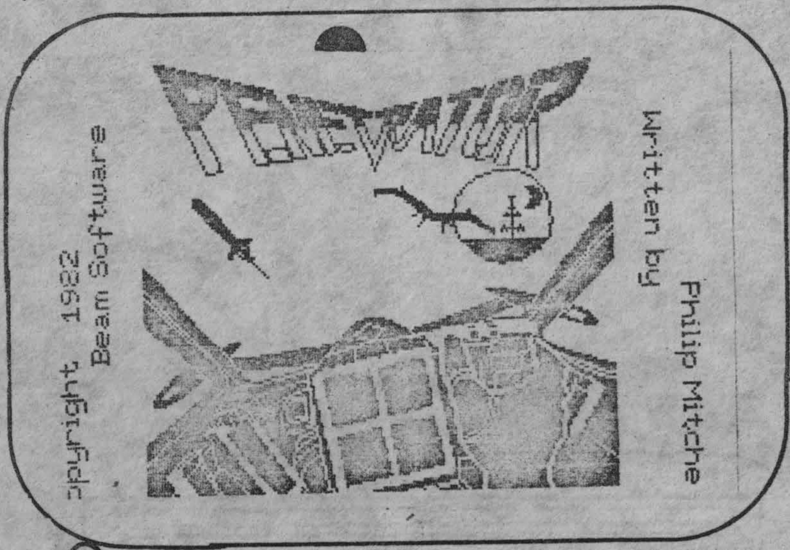
O SCLIPIRE DE INTELIGENTA

A venit timpul sa discutam despre o problema amintita anterior : spuneam ca aventurile ar trebui sa se orienteze in directia perfectionarii elementelor de inteligenta artificiala. Mai precis , nu ne referim acum la imbunatatirea interpretorului , desi si acesta este unelement de IA care concura la realizarea aventurilor. Cind vorbim despre IA , ne referim la un aspect mult mai subtil si deosebit de interesant.

Si vom arunca inca odata o privire retrospectiva. Majoritatea aventurilor , inclusiv toate cele generate in programele specializate QUILL si G.A.C. , se caracterizeaza printr-o oarecare pasivitate a personajelor , in sensul ca acestea nu dau dovada de "discernamint" si in plus au psihologii absolute. Eroii pozitivi sint intotdeauna buni , iar eroii negativi nu au alt scop decit de a pune piedici eroilor pozitivi , indiferent de situatii si interese. Mai mult , toate personajele se comporta fara "personalitate". Eroul este manipulat de catre jucator dupa bunul plac al acestuia , iar celelalte personaje care participa la actiune nu sint capabile de decizii si de regula ramin neclintite in aceeasi locatie. Aventurile din aceasta categorie







devin in scurta vreme plictisitoare , putind fi comparate cu un joc de sah la care adversarul muta intotdeauna identic.

Implementarea IA in aventuri (SPECTRUM) a inceput de la programul firmei LEGEND - VALHALLA. Desi era doar o incercare timida , VALHALLA a adus o serie de inovatii interesante. In primul rind , a fost prima aventura multi-personaj . In al doilea rind , personajelor (impartite in prieteni , inamici si monstri) li s-au conferit atribute de bunatate , magie ,forta si curaj , insuficient corelate insa.

Ideile testate in VALHALLA au fost imbunatatite in programele firmei MELBOURNE HOUSE - THE HOBBIT si SHERLOCK. In THE HOBBIT , personajele incep sa capete un gen de independenta. Pe baza unor algoritmi elaborati , ele se "misca" prin labirintul locatiilor , conform interesului lor , avid chiar si un anumit domeniu de autonomie. De exemplu , spiridusii arareori se vor aventura pina in zona padurilor. In acest program apar si primele atribute psihico-fizice coerente. Magicianul Gandalf , dispunind de o forta psihica deosebita , il poate domina pe micutul hobbit fiind capabil de exemplu de a-l deposeda pe acesta de unele obiecte. Unele personaje sint "prietenele" micului Bilbo , iar altele sint "dusmanele" sale. Oricum , aceste atribute nu sint absolute. Bilbo , incercind de exemplu sa il omoare pe prietenul sau , viteazul Thorin , il transforma pe acesta intr-un redutabil inamic , care nu va scapa nici o ocazie pentru a-si lua revansa. O alta inovatie a programului este "comunicarea" intre personaje (ANIMTALK). Bilbo ii poate cere lui Elrond sa ii descifreze harta , daca se afla cu acesta intr-un raport psihic favorabil. O scena tipica este cea a intemnitarii lui Bilbo de catre spiridusi. Prea mic pentru a ajunge la fereastra , el poate parasi temnita intr-un singur mod : cerindu-i unui prieten (Thorin) sa il ia pe umeri si apoi sa iasa pe fereastra (sintem siguri ca aceasta scena a pus mari probleme celor ce au jucat acest joc).

SHERLOCK continua ideile lui THE HOBBIT si aduce o imbunatatire in plus : evenimentele se produc in timp real. Exista un Mers al trenurilor , trenurile opresc in statie la orele fixate (!!) si cerebralul Holmes nu va putea rezolva cazul dublei crime daca nu va participa la intilniri care au loc la ore dinainte stabilite.

Vom descrie in continuare tehnicile prin care vom putea atribui personajelor reactii asemanatoare celor ale fiintelor vii : teama , ura , durere , etc.

Sa apelam din nou la un exemplu : presupunem ca eroii nostri sint doi exploratori ai adincurilor , Kagan si Marla , care in cercetarile efectuate asupra unui vas scufundat , intilnesc doi monstri : Genghis si Boreel. Acesti monstri sint inceti , dar poseda capacitati mentale excelente. Ei nu se vor angaja in lupta decit daca vor avea mari sanse de reusita. In plus Genghis poseda o sabie gasita pe vas , ceea ce il face foarte periculos.

In descrierea fiecarui personaj pot fi fi folosite unele atribute primare , care definesc starea fizico-psihica a persona-

jului la un moment dat. Sa presupunem ca atributele pe care dorim sa le folosim sint :

ATRIBUTUL	MODIFICABIL	DOMENIUL (+10/-10)
Emotional	Da	stabil/instabil
Stres	Da	singe rece/panica
Forta	Da	puternic/slab
Inteligenta	Nu	geniu/idiot
Ego	Da	dezvoltat/nedezvoltat

Bineinteles ca aceste atribute nu sint general valabile. Unele aventuri vor considera inutile unele dintre ele , in schimb vor considera necesare altele , cum ar fi : Dexteritate , Capacitate de atac (pe care o putem defini ca fiind numarul maxim de lovituri pe care personajul le poate primi fara a fi ranit) , Magie , etc.

Discutind de exemplu despre stres , sa remarcam ca daca un personaj ar avea atributul de stres la valoarea -5 , urmatoarea situatie in care intervine stresul ar decurge negativ pentru acest personaj. El va face greseli si ii vor fi afectate atributele de forta , ego si emotional.

Secretul IA folosind atributele ca si senzori ai abilitatii personajelor sta in felul in care fiecare atribut este corelat cu celalalte. Folosind acest sistem , nici un personaj nu va fi bun sau rau in mod inerent. Devine posibila chiar recrutarea unor inamici sau transformarea unor prieteni in inamici.

Sa presupunem exemplul nostru si o serie de atribute predefinite :

ATRIBUTUL	KAGAN	MARLA	BOREEL	GENGHIS
Emotional	5	-2	2	-3
Stres	4	-1	-3	6
Forta	3	2	4	4
Inteligenta	7	6	6	6
Ego	6	-5	4	7

Pe baza atributelor primare pot fi create atribute noi , mult mai complexe , pe baza unor simple formule de calcul.

ATRIBUT COMPLEX	ATRIBUTE PRIMARE IMPLICATE
Furie	Stres/Inteligenta/Ego
Curaj	Stres/Forta/Ego
Fericire	Stres/Forta/Ego
Multumire	Stres/Inteligenta/Ego



Personajelor li se poate conferi si o anumita doza de autonomie , in sensul de a decide singure asupra unor actiuni. In acest fel , eroul poate deveni rebel , ceea ce provoaca o modificare radicala a tacticii de joc.

Sa incercam acum o implementare BASIC a celor expuse mai sus.

```
  10 REM
  20 REM PROGRAMUL BASIC 1
  30 REM
  40 LET c$=""
     LET p$=""
     LET n$=""
  50 REM BANCA DE ATRIBUTE
  60 DIM a(4,6)
     REM (indice personaj, indic
e atribut)
  70 RESTORE 130
  80 FOR k=1 TO 4
  90    FOR n=1 TO 6
 100       READ a(k,n)
 110    NEXT n
 120 NEXT k
 130 DATA 5,4,3,7,6,6
     REM Kagan
 140 DATA -2,-1,2,6,-5,-6
     REM Marla
 150 DATA -5,6,4,6,7,3
     REM Genghis
 160 DATA 2,-3,4,6,4,6
     REM Boreel
 170 REM SCENARIU EXEMPLIFICATIV
 180 CLS
 190 PRINT " Apa devine din ce i
n ce mai clara pe masura ce Kaga
n si Marla  se avinta in adincur
i catre   umbrele intunecate car
e mascheaza iahtul."
 200 REM INITIALIZARE MONSTRI
 210 IF a(3,2) OR a(3,3) OR a(3,
5)<-1 THEN LET c$="Boreel"
       GO TO 240
 220 IF a(4,2) OR a(4,3) OR a(4,
5)<-1 THEN LET c$="Genghis"
       GO TO 240
 230 IF c$="" THEN PRINT "Iahtul
 se gaseste inintea exploratoril
or si nimic nu ii impiedica sa i
l cerceteze. Fara a pierde timpu
l, ei pasesc in intunericul de n
epatruns."
       STOP
 240 REM TIPARIRE MONSTRII
 250 PRINT " Dupa scurt timp, si
lueta  iahtului se desprinde din
 intuneric."'c$;" pazeste intrar
ea."
 260 REM DECIZIE DE LUPTA
 270 IF a(1,2) AND a(1,3) AND a(
1,5)<-8 THEN LET p$="Marla"
       LET n$="Kagan"
       GO TO 300
 280 IF a(2,2) AND a(2,3) AND a(
2,5)<-8 THEN LET p$="Kagan"
       LET n$="Marla"
       GO TO 300
 290 LET p$=" Kagan si Marla se
hotaresc sa  lupte."
     PRINT p$
     STOP
 300 REM UNUL DINTRE EROI FUGE
 310 PRINT " Eventualitatea lupt
ei ii ingheata lui ";n$;" singel
e in vine. El inoata spre supraf
ata, lasindu-l pe ";p$;" sa faca
 fata singur monstrilor."
     STOP
```

Presupunind ca unul dintre monstri are puternice resurse fizice si psihice , in mod inevitabil se va ajunge la lupta.

Lupta este o problema delicata , intrucit simularea ei necesita stapinirea unor tehnici speciale. Folosind IA putem face ca interactiunea dintre personaje sa fie mai realista , iar rezultatul luptei sa fie dependent de personaje si nu de circumstante.

Primul lucru care trebuie testat este daca lupta e sau nu dezirabila. Apoi trebuiesc stabilite conditiile interactiunii : cine este implicat in lupta si daca unul sau mai multe personaje sint avantajate (poseda arme sau obiecte magice). Abia dupa aceste verificari poate incepe lupta. Fiecare faza a luptei este dependenta de fazele anterioare. Exista doua aspecte de IA de care trebuie tinut cont inainte de fiecare secventa de lupta : primul este Modelul Evenimentelor Limitate (MEL) , un aspect legat de psihologie. Cel de-al doilea il reprezinta construirea unor formule care sa asigure o lupta corecta.

MEL pleaca de la psihologia anticiparii. In cadrul unei aventuri , se asteapta ca jucatorul sa ia ANUMITE decizii. Subrutina de lupta va lua in considerare urmatorii factori :

1) Eroul (eroii) s-au angajat in lupta ?

2) Citi inamici sint angajati in lupta ?
3) Raportul fortelor intre personaje.
4) Este cineva inarmat ?
5) Modalitatile de a abandona lupta.
6) Consecintele victoriei.

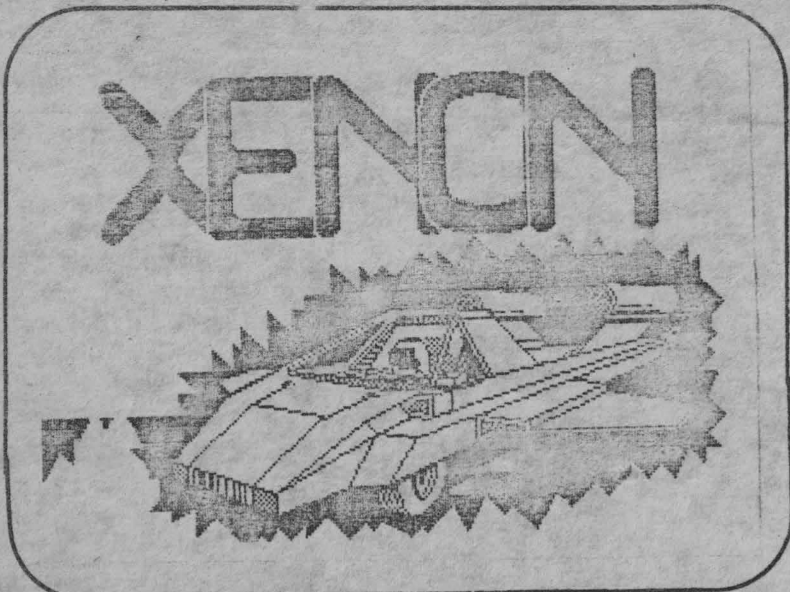
Raportul fortelor depinde puternic de atributele personajelor. Dupa cum spuneam , pe baza atributelor primare pot fi create atribute noi , complexe :

ATRIBUTUL	KAGAN	MARLA	BOREEL	GENGHIS
Ego	5	9	9	6
Furie	3	2	7	7
Curaj	7	-4	-2	-3
Inteligenta	9	9	-7	-7
Forta	4	-3	7	5
Agilitate	5	9	-3	-3

Kagan este un bun luptator , neavind atribute negative. Marla , in schimb , nu va face fata deoarece este infricosat si extenuat.

Continuam cu un nou program exemplificativ care va simula o scena de lupta.

```
 10 REM
 20 REM PROGRAMUL BASIC 2
 30 REM
 40 REM SCENA LUPTEI. CADRUL 1.
 50 LET n$=""
 60 RESTORE 130
 70 DIM c(4,6)
 80 FOR k=1 TO 4
 90   FOR n=1 TO 6
100     READ c(k,n)
110   NEXT n
120 NEXT k
130 DATA 5,3,7,9,4,3
    REM Kagan
140 DATA 9,2,-4,9,-3,4
    REM Marla
150 DATA 9,7,-2,-7,7,-3
    REM Boreel
160 DATA 6,7,-3,-7,5,-3
    REM Genghis
170 IF c(1,5)<-9 THEN PRINT " Kagan nu mai poate continua."
    LET n$="Kagan"
    GO SUB 270
    REM KAGAN E MORT?
180 IF c(2,5)<-9 THEN PRINT " Marla nu mai poate continua."
    LET n$="Marla"
    GO SUB 270
    REM MARLA E MORT?
190 LET g$=""
200 IF c(3,5)<-9 OR c(4,5)<-9 THEN GO SUB 330
    REM MONSTU UCIS
210 IF g$="STOP" THEN STOP
220 IF c(1,2)< c(3,2) AND c(1,3) > c(3,3) AND c(1,5) < c(3,5) AND c(1,6) < c(3,6) THEN LET c(1,2)=c(1,2)+INT (RND*2)
    LET c(1,5)=c(1,5)-INT (RND*2)
    GO TO 170
230 LET c(3,2)=c(3,2)+INT (RND*2)
    LET c(1,5)=c(1,5)-INT (RND*2)
240 IF c(2,2)< c(4,2) AND c(2,3)> c(4,3) AND c(2,5)< c(4,5) AND c(2,6)< c(3,6) THEN LET c(2,2)=c(2,2)+INT (RND*2)
    LET c(2,5)=c(2,5)-INT (RND*2)
    LET c(2,6)=c(2,6)-INT (RND*2)
    GO TO 170
250 LET c(4,2)=c(4,2)+INT (RND*2)
    LET c(4,5)=c(4,5)-INT (RND*2)
    LET c(4,6)=c(4,6)-INT (RND*2)
260 GO TO 170
270 REM MONSTRUL UCIDE UN EROU
280 IF c(3,5)> 4 AND c(3,6)> 5 AND c(3,3)> 4 THEN PRINT " Boreel il ucide pe ";n$
    LET n$=n$+"x"
290 IF c(4,3)>4 AND c(4,6)>4 AND c(4,5)>6 THEN PRINT " Genghis il omoara pe ";n$
    LET n$=n$+"x"
300 IF n$="Kaganx" THEN LET c(2,1)=c(2,1)-2
    LET c(2,2)=c(2,2)+2
    REM Atributele lui Marla se modifica la moartea lui Kagan
310 IF n$="Marlax" THEN LET c(1,1)=c(1,1)-2
    LET c(1,2)=c(1,2)+2
320 RETURN
330 REM EROII AU UCIS UN MONSTRU
340 IF c(3,5)<-9 THEN PRINT " Cu un geamat prelung, Boreel se prabuseste secerat. Boreel e mort."
350 IF c(4,5)<-9 THEN PRINT " Cu o lovitura puternica, cei doi cercetatori il ucid pe Genghis"
    LET g$="STOP"
360 RETURN
```



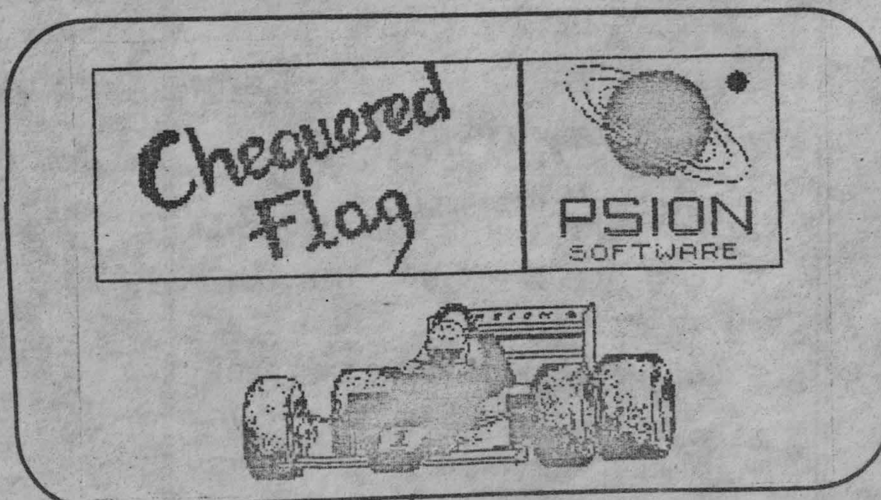
Mai exista un aspect al problemei pe care l-am amintit , dar pe care nu l-am luat inca in considerare. Este vorba despre sabia din posesia lui Genghis. Conform MEL , exista doar doua posibilitati de a procura acest obiect. O posibilitate este de a ucide amindoi monstrii , dupa care sabia devine disponibila. Cea de-a doua posibilitate este ca unul dintre eroi sa creeze o diversiune , timp in care celalalt sa profite de confuzia creata si sa ii subtilizeze monstrului sabia.

Problema sabiei poate fi simulata foarte usor in exemplul anterior , adaugind in program citeva linii care sa testeze agilitatea si forta eroilor pozitivi. Daca unul dintre ei are valori crescute (peste +5) pentru aceste atribute , el va incerca o diversiune , timp in care celalalt erou va fura sabia. Evident ca avind in posesie aceasta arma , monstrii vor fi mult mai usor de invins.

Nu vom mai insista acum asupra IA , subiectul fiind foarte larg. Lasam pe fiecare cititor sa isi imagineze noi mijloace de imbunatatire a tehnicilor prezentate , noi atribute definitorii si mai ales relatii intre ele.

In numarul urmator al buletinului vom continua articolul nostru , prezentind principala sectiune a sistemului nostru : evaluatorul , impreuna cu modul de construire a bazelor de date ale personajelor , considerente privind executia unui sistem care sa functioneze in timp real , precum si mijloace atractive de imbunatatire a graficii si sunetului.

teeb & draghi
martie 1989



MODIFICAREA SETULUI DE CARACTERE LA MICROCALCULATOARELE COMPATIBILE CU SINCLAIR ZX SPECTRUM

○ Dan Magiaru ○

CUPRINS :

1. INTRODUCCERE

Probabil ati vazut jocuri sau programe utilitare la care mesajele sint afisate cu niste caractere diferite de cele pe care le afisaza calculatorul in mod obisnuit. In cele ce urmeaza, veti vedea cum puteti realiza acest lucru. Desi s-ar putea sa va para complicat, de fapt, lucrurile sint simple, iar dupa ce reusiti sa alcatuiti pe o banda o " biblioteca " de seturi de caractere, folosirea lor este cit se poate de usoara.

Acest articol, destinat in primul rind incepatorilor, isi propune sa prezinte cum se pot extrage, genera si folosi seturi de caractere diferite, care sa se potriveasca mai bine la

programele pe care le-ati scris sau pe care le veti scrie, pentru a le face cit mai interesante si atragatoare. Dorinta autorului a fost ca prezentarea problemei sa fie cit mai explicita, iar nivelul cititorilor poate fi foarte variat. De aceea, autorul isi cere scuze pentru repetarea unor lucruri bine cunoscute de unii. Cuprinsul de la inceput ii va ajuta sa sara peste ceea ce ii intereseaza mai putin. In caz ca anumite probleme mai necesita lamuriri, autorul va sta cu placere la dispozitie.

2. STOCAREA SETULUI DE CARACTERE

a) UNDE SE FACE STOCAREA SETULUI DE CARACTERE

Setul de caractere al calculatorului se afla inscris in ROM, astfel incit el este utilizabil imediat cind punem calculatorul in functie, fara nici o pregatire prealabila.

Adresa la care incepe setul de caractere este indicata de variabila de sistem CHARS (a se vedea [1], in partea a cincea a acestui articol, la bibliografie), aflata in RAM, la adresele 23606 si 23607. In fiecare din aceste adrese se afla inscris un byte (un numar intre 0 si 255 sau, in hexazecimal, intre # 00 si # FF - semnul " # " se numeste diez si se foloseste in muzica; aici va indica un numar hexazecimal, adica in baza de numeratie 16). Cei 2 bytes, ne dau prin alaturare un numar cuprins intre 0 si 65535, sau # 0000 si # FFFF. In locatia 23606 se scrie byte-ul cel mai putin semnificativ (LSB - Least Significant Byte) iar in locatia 23607 se scrie byte-ul cel mai semnificativ (MSB - Most Significant Byte). Numarul inscris in CHARS este cu 256 mai mic decit adresa unde incepe setul de caractere. De ce ? Deoarece setul de caractere care ne intereseaza incepe cu spatiu - sau blanc - (CHR$ 32) si se termina cu " @ " (CHR$ 127). Dupa cum vom vedea mai jos, un caracter este format din 8 bytes. Primele 32 de caractere (de la 0 la 31), care nu ne intereseaza aici, ocupa 32 * 8 bytes, deci 256 bytes.

La pornirea calculatorului, in locatia 23606 este inscris 0 iar in locatia 23607 este inscris 60.

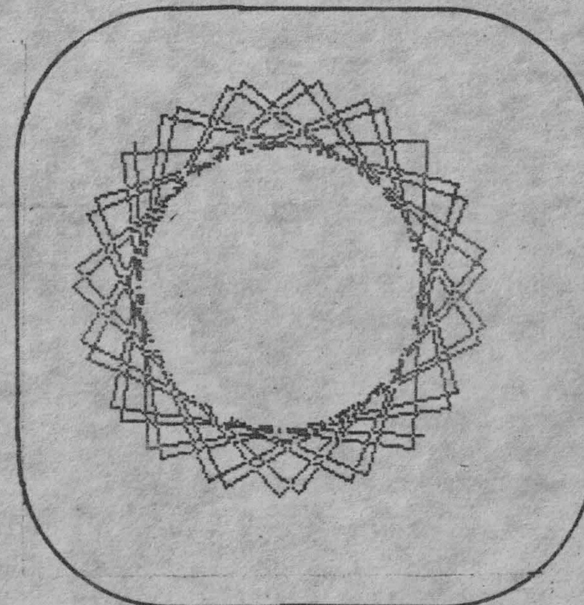
Astfel, pentru a afla valoarea variabilei de sistem CHARS, introducem comanda :

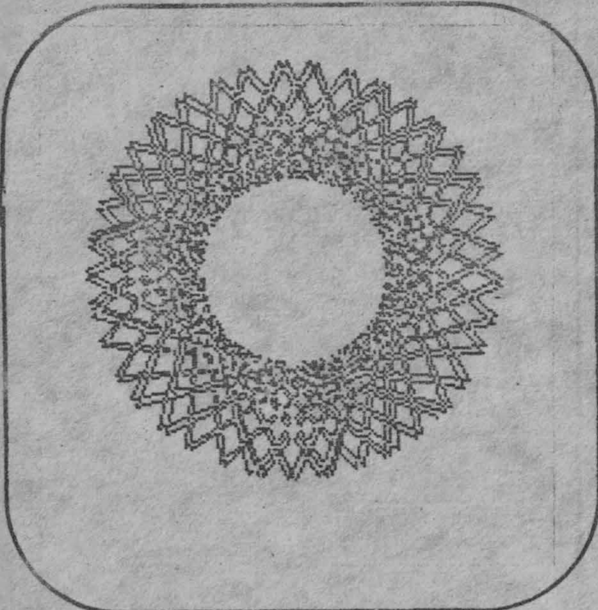
```
        PRINT PEEK 23606 + 256 * PEEK 23607

adica           ( 0        ) + ( 256 * ( 60  ))
```

Vom obtine rezultatul 15360. Adaugind 256, obtinem 15616, care reprezinta adresa de inceput a setului de caractere.

Setul de caractere are o lungime de 768 (# 0300) bytes, adica 96 caractere * 8 bytes.





?

b) CUM SE FACE STOCAREA SETULUI DE CARACTERE

In mod normal, deci cind se scrie cu caractere de dimensiuni obisnuite, adica pe 22 / 24 de linii si 32 de coloane, fiecare caracter este reprezentat printr-o matrice de 8 * 8 pixeli. In aceasta matrice, fiecare linie este reprezentata printr-un byte, in care fiecare bit reprezinta un pixel. Daca un anumit bit are valoarea 1, pixel-ul respectiv va avea culoarea INK; daca bit-ul este 0, pixel-ul va avea culoarea PAPER. Cei 8 bytes reprezinta cele 8 linii ale matricii de sus in jos.

De exemplu, caracterul " A " se reprezinta prin urmatorii 8 bytes: 0 60 66 66 126 66 66 0 . Iata cum:

prima	linie =	0	( )	0 0 0 0 0 0 0 0
a doua	linie =	60	(32+16+8+4)	0 0 1 1 1 1 0 0
a treia	linie =	66	(64+ 2)	0 1 0 0 0 0 1 0
a patra	linie =	66	(64+ 2)	0 1 0 0 0 0 1 0
a cincea	linie =	126	(64+32+16+8+4+2)	0 1 1 1 1 1 1 0
a sasea	linie =	66	(64+ 2)	0 1 0 0 0 0 1 0
a saptea	linie =	66	(64+ 2)	0 1 0 0 0 0 1 0
a opta	linie =	0	( )	0 0 0 0 0 0 0 0

Dupa cei 8 bytes, urmeaza altii 8, care definesc un alt caracter, (in cazul de fata " B "), si asa mai departe.

3. INREGISTRAREA DE PE BANDA MAGNETICA

Pe banda magnetica, un set de caractere poate fi stocat sub forma unui fisier de tip " Byte ", salvat la adresa " adr " si lung de 768 bytes.

Sa presupunem ca avem deja pe banda un fisier cu un set de caractere diferit de cel uzual. Inca nu am discutat cum il putem obtine; o vom face mai tirziu, in partea a patra a acestui articol; acest salt l-am facut pentru o mai usoara intelegere a problemei.

Putem inregistra in RAM noul set de caractere efectuind urmatoarele operatii:

1. Alegem un numar de locatie in RAM; il vom numi " adr " .
2. Pentru a proteja noul set de caractere,de stergerea prin " NEW ", vom cobori RAMTOP-UL la (adr - 1), cu

```
CLEAR adr - 1
```

3. Incarcam noul set de caractere in memorie cu:

```
LOAD "" CODE adr,768
```

Adr poate fi diferit de adresa la care a fost salvat

fisierul pe banda. Astfel, nu trebuie sa ne ingrijoreze faptul ca nu cunoastem sau ca nu ne convine adresa la care a fost salvat fisierul. Acesta este relocatabil.

Dupa ce am efectuat aceste operatii, avem noul set in RAM. Totusi, calculatorul scrie in continuare cu caracterele din ROM. Aceasta deoarece CHARS a ramas la valoarea initiala.

Pentru a scrie cu noile caractere, trebuie sa modificam valoarea variabilei de sistem CHARS, pentru ca aceasta sa indice adresa noului set.

Trebuie sa aflam valoarea zecimala a celor 2 bytes. Vom proceda astfel:

```
LET nn  = adr - 256 : REM nn = noua valoare a lui CHARS
LET MSB = INT ( nn / 256 )
LET LSB = nn - 256 * MSB
```

Inscriem acum aceste valori la adresele 23606 si 23607:

```
POKE 23606 , LSB
POKE 23607 , MSB
```

De acum calculatorul va folosi noul set de caractere in locul celui obosnuit si asta fara sa intram in modul grafic.

Revenirea la caracterele originale se face foarte simplu:

```
POKE 23606 , 0
POKE 23607 , 60
```

Setul de caractere original fiind in ROM, se pastreaza tot timpul, putind fi astfel apelat oricind. Aceasta revenire nu afecteaza cu nimic setul de caractere introdus de noi. Cu:

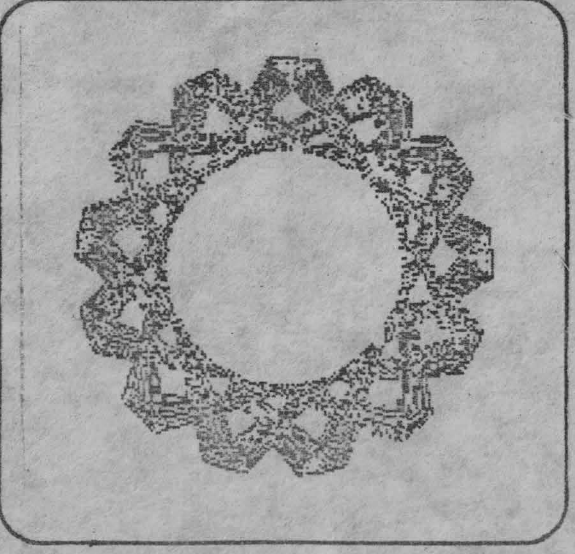
```
POKE 23606 , LSB : POKE 23607 , MSB
```

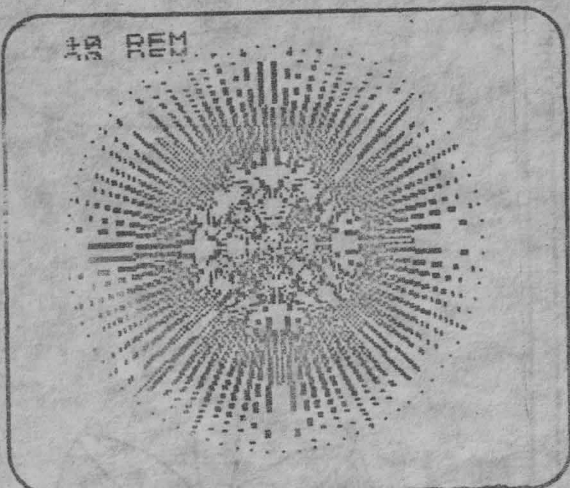
il vom obtine din nou pe acesta din urma.

Iata cum ar putea arata un exemplu, presupunind (din nou) ca avem banda pregatita.

```
 10 INPUT "adr = " ; adr : REM adr = adresa noului set
 20 CLEAR adr - 1 : REM coborim RAMTOP-UL
 30 LET nn = adr - 256 : REM nn =  noua  valoare  a  lui
    CHARS
 40 LET MSB = INT ( nn / 256 )
 50 LET LSB = nn - 256 * MSB
 60 PRINT " LSB " ; LSB
 70 PRINT " MSB " ; MSB
 80 LOAD "" CODE adr,768 : REM  putem,  eventual,  scrie
    intre ghilimele numele fisierului (setului de
    caractere ) de pe banda
 90 POKE 23607 , MSB
100 POKE 23606 , LSB
```

Asta este tot ! Notam LSB si MSB, deoarece, dupa NEW, ele trebuiesc reintroduse prin POKE.





Valoarea lui adr o vom lua destul de sus, pentru a lasa cit mai mult loc pentru programul BASIC sau in Cod-Masina.

O valoare convenabila este 64256, deoarece :

- inscrie in CHARS numarul nn = 64000, deci:
- LSB = 0
- MSB = 250

Ori, LSB = 0 si in cazul setului de caractere original, iar MSB are o valoare usor de retinut (250).

Un ultim sfat:
Putem incarca mai multe seturi de caractere in acelasi program, la adrese diferite si sa le apelam apoi dupa dorinta. In acest caz, evident ca vom cobori RAMTOP-ul sub valoarea cea mai mica a lui adr.

4. OBTINEREA UNUI SET DE CARACTERE

a) OBTINEREA UNUI SET DE CARACTERE PRIN GENERARE

Pentru a genera un set de caractere, putem folosi un program specializat, cum ar fi " CARASET ", program scris in BASIC, care genereaza 9 seturi de caractere, pe care le salveaza apoi la adresa 64256. Modificindu-l, vom putea obtine si alte seturi.

Putem, de asemenea scrie un program care sa creeze automat un set de caractere modificind setul din ROM si sa-l stocheze apoi in RAM. Asa am scris " Gen. it. CHR$ ", program care genereaza caractere italice (inclinate). Acesta are la baza urmatorul principiu:

- ia valorile continute la adresele incepind cu 15616 si terminind cu (15616 + 768), pe grupuri de cite 8 bytes (ia cite un caracter);
- imparte primii bytes la 2 (muta pixelii la dreapta);
- inmulteste ultimii bytes cu 2 (muta pixelii la stinga);
- ii lasa neschimbati pe cei din mijloc;
- ii stocheaza in locul ales din RAM;
- continua cu urmatorul caracter, pina termina intregul set.

Mai convenabila este, insa, utilizarea unui program grafic, cum ar fi " The Artist II ", " LIGHTMAGIC ", " artstudio ", etc.

Recomandam cu caldura " artstudio ", scris de James Hutchby, OCP, 1985, autor al excelentului program de biliard " Video Pool ". " artstudio ", scris intr-o maniera moderna este foarte usor de folosit datorita existentei unui numar mic de taste de comanda (stinga, dreapta, sus, jos si " foc "), marele numar de comenzi posibile fiind apelate prin controlarea unui cursor-sageata pe meniul afisat permanent in partea superioara a ecranului. Pentru a genera cu ajutorul sau un set de caractere, intrati in modul " Text ", apoi ,din acesta, in modul " Font ". In final, fara a iesi din modul " Font ", intrati in modul " File " si salvati setul de caractere nou creat.

b) OBTINEREA SETURILOR DE CARACTERE DIN JOCURI

Pentru a obtine pe banda setul de caractere dintr-un program (de obicei dintr-un joc), trebuie doar sa intrerupem rularea acestuia cu " BREAK ", pentru a afla adresa unde incepe setul de caractere. Apoi, cu :

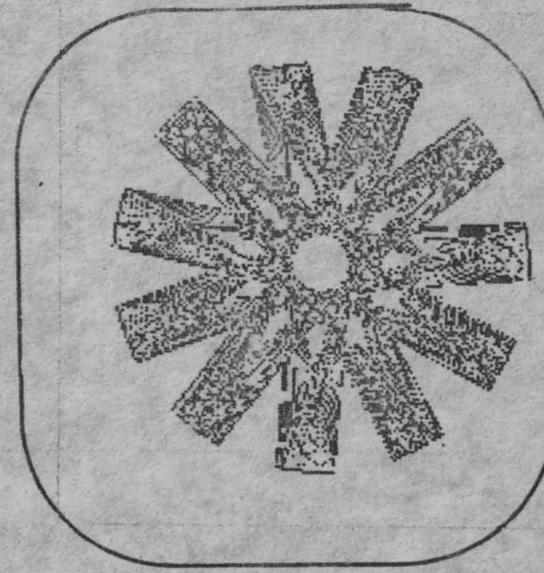
```
        PRINT PEEK 23606   aflam LSB
iar cu  PRINT PEEK 23607   aflam MSB
        CHARS = 256 * MSB + LSB
iar     adr = 256 + CHARS
```

Astfel se pot obtine seturile de caractere " FONT_1 " si " FONT_2 " din MEGA BASIC [2].

Din pacate, lucrurile nu sint intotdeauna atit de simple ! Majoritatea programelor scrise in Cod-Masina nu se pot BREAK-a, fiind protejate. Unele pot fi " pacalite ", de obicei utilizind optiunile de salvare / incarcare a scorului. Astfel se poate obtine setul de caractere din programul muzical " WHAM! ".

Daca protectia este mai " puternica " decit noi, va trebui sa folosim un program utilitar specializat. Un program foarte util este " SPRITE BUSTER ", scris de ing. Miodrag Puterity [3]. In acest articol este listat programul in cod-masina, loader-ul BASIC, precum si manualul de utilizare al acestui program.

Pentru afisarea cu caractere mai mari decit 8 * 8 pixeli, recomandam articolul semnat de cercet. Ion Diamandi, ITCI, in revista " Stiinta si tehnica " [4], in care este folosit setul de caractere din ROM. Modificind corespunzator valoarea atribuita variabilei A in linia 10, se poate folosi un alt set de caractere.



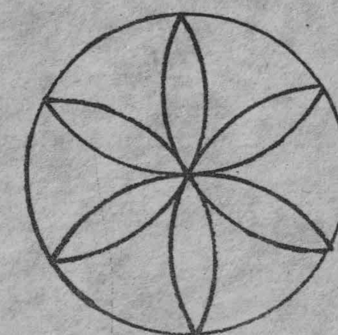
5. BIBLIOGRAFIE

[1] Steven Vickers -" Sinclair ZX Spectrum BASIC Programming "
1982 Sinclair Research Ltd.
25, Willis Rd, Cambridge
editia a treia, 1983, cap. 25

[2] Mike Leaman - " MEGABASIC - Your Spectrum 1985 "
in INF nr. 1 - 2 / 1987, pag. 53
Buletin al Clubului Programatorilor
Casa Universitarilor Timisoara
Str. Paris nr. 1
1900 Timisoara

[3] ing. Miodrag Puterity - " Program pentru vizualizat sprite-uri, seturi de caractere si u.d.g.-uri "
in INF nr. 1/88, pag. 82 - 92

[4] Ion Diamandi - " O subrutina pentru marirea caracterelor afisate "
in " Stiinta si tehnica " nr. 2 / 1989, pag.26



LIMBAJUL MICRO-PROLOG IN APLICATII

• Kecskemeti Nicolae •

Programul de fata, scris in limbajul micro-PROLOG,este un microsistem bazat pe cunostinte, destinat identificarii unor animale pe baza intrebarilor puse utilizatorilor. Microsistemul este dotat si cu un mecanism de invatare din exemple.Nu se pretinde a fi un sistem expert propriu-zis, ci ilustreaza doar principiile de baza dupa care se reprezinta si se gestioneaza cunoasterea in sistemele expert.

Programul se imparte in:

1. Baza de cunostinte, reprezentata de clauzele <u>este animal</u> si <u>este</u>.
2. Mecanismul de generare al intrebarilor si de retinere al raspunsurilor,format din clauzele <u>pozitiv</u>, <u>negativ</u>, <u>intreaba</u> si <u>retine</u>.
3. Mecanismul de invatare, format din clauzele <u>deosebeste</u>, <u>modifica</u>, <u>stabileste clasa</u>, <u>stabileste categoria</u>, <u>cere date</u>, <u>cere</u>, <u>completeaza</u>, <u>completn</u> si <u>genereaza</u>.

Lansarea programului se face tastind <u>start</u> <u>a</u>. Din acest moment, sistemul pune intrebari privind animalul la care s-a gindit utilizatorul, la care utilizatorul raspunde cu <u>da</u> sau <u>nu</u>. Cind ansamblul raspunsurilor corespunde unui animal din baza de cunostinte, calculatorul indica acest animal, cerind apoi confirmarea utilizatorului (<u>da</u>/<u>nu</u>). La raspuns afirmativ, programul se opreste. La raspuns negativ, sistemul cere un element distinctiv al animalului indicat, fata de cel la care s-a gindit utilizatorul. Acesta trebuie introdus sub forma de lista: intre paranteze, doua siruri de caractere separate prin spatiu, eventual precedate de <u>nu</u> urmat de spatiu. Exemple: (cinta frumos) (nu se_hraneste_cu carne). Calculatorul adauga aceasta proprietate la regula referitoare la animalul pe care l-a identificat, apoi cere numele animalului la care s-a gindit utilizatorul. Genereaza o noua regula pentru acest animal si de acum inainte va putea recu-

noaste si acest animal (si-a imbogatit baza de cunostinte).

Analog creste baza de cunostinte in cazul in care din raspunsurile date, calculatorul nu poate identifica nici un animal.

<u>Bibliografie</u>:

[1] Turbo PROLOG Owners Handbook, Borland International, 1986

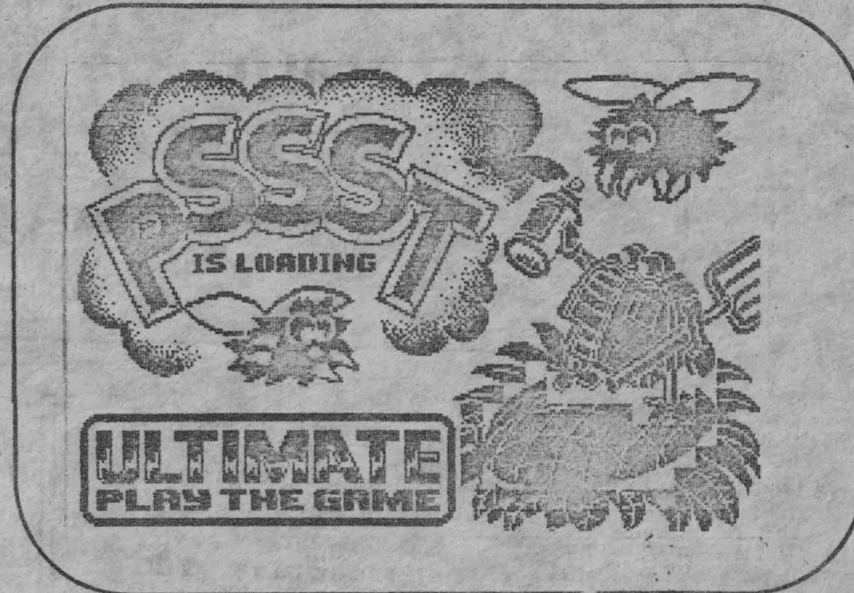
[2] Revista INF nr.1,Casa Universitarilor Timisoara, 1988, pag.61-70.

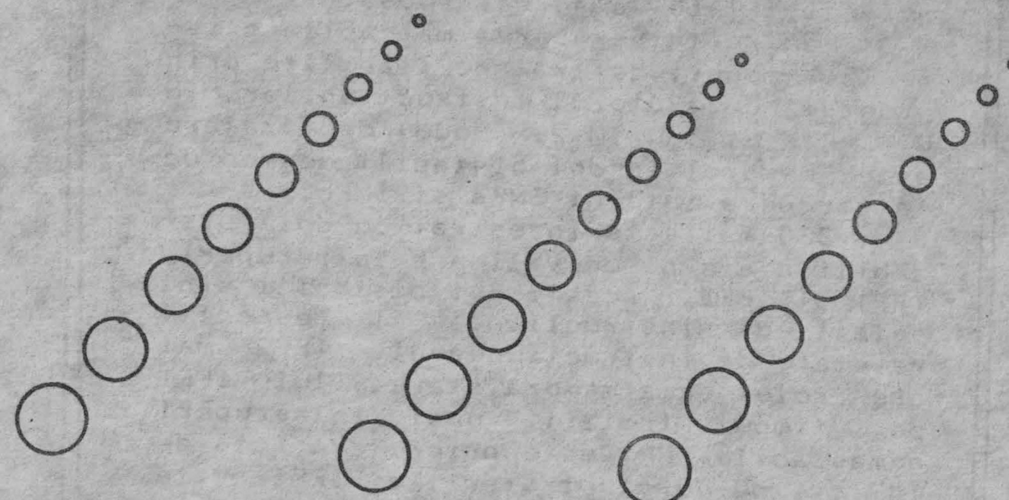
```
((genereaza X)
  (/* Genereaza clauza)
  (ADDCL X))
((retine X Y da)
  (/* Retine caracteristici din
raspuns)
  (ADDCL ((apozitiv X Y))))
((retine X Y nu)
  (ADDCL ((anegativ X Y))))
((completn X Y Z)
  (/* Completeaza raspunsuri neg
ative)
  (EQ x anegativ)
  (CL ((x|y)|z))
  (EQ X1 (x))
  (adauga X1 y Y1)
  (NOT membru Y1 Y)
  (EQ Z1 negativ)
  (EQ x1 (Z1))
  (adauga x1 y y1)
  (adauga X (y1) Z)
  (adauga (Y1) Y z1)
  (completn Z z1 X2))
((completn X Y X)
  (genereaza X))
((membru X (X|Y)))
((membru X (Y|Z))
  (/* Apartenenta la o lista)
  (membru X Z))
((intreaba X Y da)
  (/* Formuleaza intrebarea)
  (P "@M" X Y "?@M")
  (R Z)
  (EQ Z da)
  (/)
  (retine X Y da))
((intreaba X Y da)
  (retine X Y nu)
  FAIL)
((intreaba X Y nu)
  (P "@M" X Y "?@M")
  (R Z)
  (EQ Z nu)
  (/)
  (retine X Y nu))
((intreaba X Y nu)
  (retine X Y da)
  FAIL)
((completeaza X Y Z x)
  (EQ y apozitiv)
  (CL ((y|z)|X1))
  (EQ Y1 (y))
  (adauga Y1 z Z1)
  (NOT membru Z1 Y)
  (EQ x1 pozitiv)
  (EQ y1 (x1))
  (adauga y1 z z1)
  (adauga X (z1) x)
  (adauga (Z1) Y X2)
  (completeaza x X2 Z Y2))
((completeaza X Y Z x)
  (completn X Z x))
((cere X altceva altceva Y Z x)
  (completeaza Y Z x y))
((cere X altceva Y Z x y)
  (EQ z (este Y))
```

```
  (adauga Z (z) X1)
  (completeaza X1 x y Y1))
((cere X Y altceva Z x y)
  (EQ z (este Y))
  (adauga Z (z) X1)
  (completeaza X1 x y Y1))
((cere X Y Z x y z)
  (/* Adauga clasa si categoria
in clauza)
  (EQ X1 (este Y))
  (adauga x (X1) Y1)
  (EQ Z1 (este Z))
  (adauga Y1 (Z1) x1)
  (completeaza x1 y z y1))
((start a)
  (/* Se pun intrebarile)
  (este_animal X)
  (/)
  (P "@M Probabil ca v-ati gindi
t la @M" X "@M")
  (PP Am dreptate ?)
  (R Y)
  (deosebeste X Y)
  (sterge_fapte))
((start a)
  (P "@M Nu pot determina animal
ul dv.")
  (P "@M Rog dati numele animalu
lui dv. @M")
  (R X)
  (stabileste_clasa X)
  (/)
  (sterge_fapte))
((este_animal pinguin)
  (/* Animalul este pinguin daca
  este pasare , nu stie sa zboare
  stie sa inoate si are culoare a
lba si neagra)
  (este pasare)
  (negativ stie_sa zboare)
  (pozitiv stie_sa inoate)
  (pozitiv are culoare_alba_si_n
eagra))
((este_animal strut)
  (este pasare)
  (negativ stie_sa zboare)
  (pozitiv are picioare_lungi)
  (pozitiv are culoare_alba_si_n
eagra))
((este_animal tigru)
  (este mamifer)
  (este carnivor)
  (pozitiv are culoare_maro_desc
his)
  (pozitiv are dungi_negre))
((este_animal girafa)
  (este copitat)
  (pozitiv are git_lung)
  (pozitiv are picioare_lungi)
  (pozitiv are pete_negre))
((este_animal zebra)
  (este copitat)
  (pozitiv are dungi_negre))
((este_animal leopard)
  (este mamifer)
  (este carnivor)
  (pozitiv are culoare_maro_desc
his)
  (pozitiv are pete_negre))
((este_animal albatros)
  (este pasare)
  (pozitiv zboara bine))
((deosebeste X da)
  (sterge_fapte))
((deosebeste X nu)
  (PP Dati ceva sub forma de lis
ta de 2 elemente distinctiv pt .
 X)
  (R Y)
  (EQ Z (este_animal X))
  (CL (Z|x))
  (/* Modifica vechea clauza)
  (modifica Y Z x)
  (PP Dati animalul dv .)
  (R y)
  (stabileste_clasa y)
  (/)
  (sterge_fapte))
((sterge_fapte)
  (/* Anuleaza raspunsurile pt .
 reluarea programului)
  (KILL apozitiv)
  FAIL)
((sterge_fapte)
  (KILL anegativ)
  FAIL)
((sterge_fapte)
  (/))
((stabileste_clasa X)
  (EQ Y apozitiv)
  (EQ Z (are blana))
  (EQ x ())
  (CL ((Y|Z)|x))
  (/* Daca are blana atunci este
 mamifer)
  (stabileste_categoria X mamife
r))
((stabileste_clasa X)
  (EQ Y apozitiv)
  (EQ Z (da lapte))
  (EQ x ())
  (CL ((Y|Z)|x))
  (stabileste_categoria X mamife
r))
((stabileste_clasa X)
  (EQ Y apozitiv)
  (EQ Z (are pene))
```

```
  (EQ x ())
  (CL ((Y|Z)|x))
  (stabileste_categoria X pasare
))
((stabileste_clasa X)
  (EQ Y apozitiv)
  (EQ Z (stie_sa zboare))
  (EQ x ())
  (CL ((Y|Z)|x))
  (EQ y (depune oua))
  (CL ((Y|y)|x))
  (stabileste_categoria X pasare
))
((stabileste_clasa X)
  (stabileste_categoria X altcev
a))
((pozitiv X Y)
  (EQ Z apozitiv)
  (EQ x (X Y))
  (EQ y ())
  (CL ((Z|x)|y))
  (/))
((pozitiv X Y)
  (EQ Z anegativ)
  (EQ x (X Y))
  (EQ y ())
  (NOT CL ((Z|x)|y))
  (/* Numai daca nu s-a mai ntre
bat)
  (intreaba X Y da))
((negativ X Y)
  (EQ Z anegativ)
  (EQ x (X Y))
  (EQ y ())
  (CL ((Z|x)|y))
  (/* S-a mai intrebat)
  (/))
((negativ X Y)
  (EQ Z apozitiv)
  (EQ x (X Y))
  (EQ y ())
  (NOT CL ((Z|x)|y))
  (intreaba X Y nu))
((adauga (X|Y) Z (X|x))
  (/* Concateneaza 2 liste)
  (adauga Y Z x))
((adauga () X X)
  (/))
((modifica (nu|X) Y Z)
  (/* Adauga in clauza elementul
 precizat)
  (adauga (negativ) X x)
  (adauga Z (x) y)
  (ADDCL (Y|y))
  (DELCL (Y|Z)))
((modifica X Y Z)
  (adauga (pozitiv) X x)
  (adauga Z (x) y)
  (ADDCL (Y|y))
  (DELCL (Y|Z)))
((cere_date X Y Z)
  (EQ x ((este_animal X)))
  (EQ y ((apozitiv are blana) (a
pozitiv da lapte) (apozitiv are
pene) (apozitiv stie_sa zboare)
(apozitiv depune oua) (apozitiv
se_hraneste_cu carne) (apozitiv
are dinti_ascutiti) (apozitiv ar
e ochi_ageri) (apozitiv are copi
te) (apozitiv rumega iarba)))
  (EQ Y mamifer)
  (EQ z ((anegativ are pene) (an
egativ stie_sa zboare) (anegativ
 depune oua)))
  (cere X Y Z x y z))
((cere_date X Y Z)
  (EQ x ((este_animal X)))
  (EQ y ((apozitiv are blana) (a
pozitiv da lapte) (apozitiv are
pene) (apozitiv stie_sa zboare)
(apozitiv depune oua) (apozitiv
se_hraneste_cu carne) (apozitiv
are dinti_ascutiti) (apozitiv ar
e ochi_ageri) (apozitiv are copi
te) (apozitiv rumega iarba)))
  (EQ Y pasare)
  (EQ z ((anegativ are blana) (a
negativ da lapte)))
  (cere X Y Z x y z))
((cere_date X Y Z)
  (EQ x ((este_animal X)))
  (EQ y ((apozitiv are blana) (a
pozitiv da lapte) (apozitiv are
pene) (apozitiv stie_sa zboare)
(apozitiv depune oua) (apozitiv
se_hraneste_cu carne) (apozitiv
are dinti_ascutiti) (apozitiv ar
e ochi_ageri) (apozitiv are copi
te) (apozitiv rumega iarba)))
  (EQ z ())
  (cere X Y Z x y z))
((este mamifer)
  (/* Este mamifer daca are blan
a)
  (pozitiv are blana))
((este mamifer)
  (pozitiv da lapte))
((este pasare)
  (pozitiv are pene))
((este pasare)
  (pozitiv stie_sa zboare)
  (pozitiv depune oua))
((este carnivor)
  (pozitiv se_hraneste_cu carne)
)
((este carnivor)
  (pozitiv are dinti_ascutiti)
  (pozitiv are ochi_ageri))
((este copitat)
  (este mamifer)
```

```
  (pozitiv are copite))
((este copitat)
  (este mamifer)
  (pozitiv rumega iarba))
((stabileste_categoria X Y)
  (EQ Z apozitiv)
  (EQ x (se_hraneste_cu carne))
  (EQ y ())
  (CL ((Z|x)|y))
  (/* Daca se hraneste cu carne
atunci este carnivor)
  (cere_date X Y carnivor))
((stabileste_categoria X Y)
  (EQ Z apozitiv)
  (EQ x (are dinti_ascutiti))
  (EQ y ())
  (CL ((Z|x)|y))
  (EQ z (are ochi_ageri))
  (CL ((Z|z)|y))
  (cere_date X Y carnivor))
((stabileste_categoria X Y)
  (EQ Z apozitiv)
  (EQ x (are copite))
  (EQ y ())
  (CL ((Z|x)|y))
  (cere_date X Y copitat))
((stabileste_categoria X Y)
  (EQ Z apozitiv)
  (EQ x (rumega iarba))
  (EQ y ())
  (CL ((Z|x)|y))
  (cere_date X Y copitat))
((stabileste_categoria X Y)
  (cere_date X Y altceva))
```





SISTEMUL DE INTRERUPERI LA Z80 CPU

• HARALD SCHRIMPF •

Microprocesorul Z80 accepta doua semnale de intrerupere: NMI, intrerupere nemascabila, si INT, intrerupere mascabila prin program.

La intreruperea nemascabila Z80 raspunde intr-un singur mod: executia rutinei aflate la adresa #0066.

Pentru intreruperea mascabila exista trei moduri de tratare stabilite prin program cu ajutorul instructiunilor IM 0, IM 1 sau IM 2. Modul de tratare este memorat in doi bistabili de mod de intrerupere interni IMFa si IMFb.

Z80 mai este inzestrat cu doi bistabili interni de validare intreruperi, IFF1 si IFF2. Acesti doi bistabili sint setati de instructiunea EI respectiv resetati de instructiunea DI. IFF2 mai are rolul de a memora starea lui IFF1 pe timpul tratarii unei intreruperi nemascabile. NMI este prioritara fata de INT si pe durata tratarii ei IFF1 este fortat pe "0" (invalidare INT). Prin memorarea starii lui IFF1 in IFF2 la sfirsitul rutinei de tratare a intreruperii nemascabile se poate reface starea lui IFF1 in doua moduri:

1.Prin executia instructiunii de revenire din intreruperea nemascabila RETN starea lui IFF2 se rescrie in IFF1.

2.La ecuatia instructiunilor LD A,I sau LD A,R starea lui IFF2 este scrisa in indicatorul de paritate si dupa testarea acestuia se poate reface starea lui IFF1 prin executia, dupa caz, a uneia din instructiunile EI sau DI.

Aceasta a doua solutie poate fi aplicata insa si in alt context. Sa presupunem ca pentru un anume motiv un sir de instructiuni trebuie executate cu intreruperile dezactivate (sau activate) nestiindu-se daca intreruperile mascabile sint validate sau invalidate, dar este necesara refacerea starii de validare sau invalidare.

Iata in continuare o rutina exemplu care ar putea rezolva aceasta problema:

```
              LD     A,R
              PUSH   AF
              DI
              CALL   RUTDI
              POP    AF
              JP     PO,AFOSTDI
              EI
AFOSTDI       etc.
              .
              .
```

unde RUTDI este rutina care trebuie executata cu intreruperile dezactivate.

Aparent aceasta rutina este scrisa corect si dupa executia rutinei RUTDI cu intreruperile dezactivate reface starea lui IFF1 (starea de validare a intreruperilor mascabile); dar nu este asa. Pentru a va convinge incercati urmatorul program:

```
              EI
LDAR          LD    A,R
              JP    PO,STOP
              JR    LDAR
STOP          punct de intrerupere soft
```

El ar trebui sa ruleze la infinit dar dupa un timp se va opri in punctul de intrerupere soft, denotind o functionare incorecta a instructiunii LD A,R (incarcarea gresita a lui IFF2 in indicatorul de paritate). La acelasi rezultat se ajunge inlocuind LD A,R cu LD A,I. Trebuie amintit ca s-a presupus tot timpul prezent a unui dispozitiv care declanseaza ciclic semnale de intrerupere.

Dupa inca ale citeva texte s-a putut trage concluzia ca daca semnalul INT devine activ in timpul executiei inastructiunilor LD A,R sau LD A,I si intreruperile sint activate indicatorul P/V se incarca eronat.

Pentru calculatoarele ZX Spectrum si compatibile aceasta eroare poate fi ocolita prin urmatoarea rutina:

```
                LD     A,R
                JP     PE,AFOSTEI
                LD     A,R
AFOSTEI         DI
                PUSH   AF
                CALL   RUTDI
                POP    AF
                JP     PO,AFOSTDI
                EI
AFOSTDI         etc.
                .
                .
                .
```

Cazul pus in discutie este urmatorul: daca sistemul a lucrat cu intreruperile activate prima instructiune LD A,R ar putea incarca indicatorul P/V gresit in cazul sosirii unui semnal de intrerupere in timpul executiei acesteia. Semnalele de intrerupere la ZX Spectrum avind o perioada de 20 ms a doua instructiune LD A,R sigur nu va fi deranjata de aparitia unei intreruperi si se va executa corect.

Constatarile de mai sus s-au facut la realizarea sistemului de operare TIM-S V2 iar solutiile adoptate pot fi urmarite in rutinele SETRAM si READMEMST.

o O O O O

LIMBAJUL DE PROGRAMARE PASCAL PENTRU CALCULATORUL TIM-S

• • • •

Materialul de fata se refera la implementarea limbajului PASCAL realizat de firma HISOFT, versiunea HP4TM1.6

HISOFT-PASCAL corespunde in linii mari limbajului de referinta, definit de N.Wirth. Avind in vedere spatiul disponibil, precum si faptul ca in literatura noastra se gasesc lucrari cu privire la limbajul PASCAL, vor fi tratate in continuare doar aspectele specifice implementarii HISOFT, a caror cunoastere este necesara folosirii cu succes a limbajului pe calculatoarele de tip SPECTRUM. Problemele tratate sint grupate in trei categorii:

- Aspecte particulare ale limbajului (fata de versiunea de referinta);
- Facilitati grafice;
- Facilitati de editare a programelor si de lucru cu caseta magnetica.

1. Particularitati ale limbajului HISOFT-PASCAL

1.1. Particularitati semantice si de sintaxa

Fata de limbajul PASCAL de referinta, se evidentiaza urmatoarele trei restrictii mai importante:

- s-a eliminat tipul FILE;
- nu se accepta articole cu variante (RECORD cu CASE);
- nu se accepta proceduri si functii ca parametru.

In plus sint remarcate, in acest paragraf, urmatoarele caracteristici:

- In declaratii de constante este permisa notatia CHR(i), introdusa in scopul de a desemna caracterele de control; astfel prin

```
CONST pg=CHR(12)
```

se introduce constanta pg, care are valoarea corespunzatoare caracterului de ordin 12 din

tabela ASCII. (Se observa ca in acest caz CHR nu desemneaza o functie standard, ca atunci cind apare in cadrul unei expresii dintr-un corp de instructiuni).

- Nu se accepta declararea unui pointer catre un tip inca nedefinit. Prin urmare secventa de mai jos, prin care, de obicei, se descriu tipuri recursive (noduri ale unor liste sau arbori), este interzisa:

```
TYPE ref=^nod
     nod=RECORD
       cheie:INTEGER;
       urm:ref
          END;
```

In schimb, este permis ca definitia unui tip sa contina pointeri la el insusi. Astfel, un nod ca cel de mai sus se descrie sub forma:

```
TYPE nod=RECORD
     cheie:INTEGER;
     urm:^nod
        END;
```

-Instructiunea CASE permite specificarea unei clauze ELSE care se executa in cazul in care valoarea expresiei selectoare nu corespunde nici uneia dintre etichete; in acest caz forma instructiunii este urmatoarea:

```
CASE 1 OF
     et_1:lista instr_1
     .
     .
ELSE:lista_instr;
```

- Avind in vedere ca nu se utilizeaza fisiere, nu au sens nici parametrii INPUT, OUTPUT in declaratia de program.

1.2. Marimi standard

1.2.1. Constante si tipuri standard.

Alaturi de constantele standard booleene TRUE si FALSE si constanta NIL, s-a introdus constanta de tip intreg MAXINT; ea are valoarea 32667 corespunzatoare celei mai mari valori intregi disponibile.

1.2.2. Proceduri si functii standard

1.2.2.1. Operatii de intrare/iesire

Procedura standard WRITE provoaca afisarea unor date la terminal sau tiparirea lor la imprimanta (o comanda de scriere de forma WRITE (CHR(16)) redirectioneaza iesirea pe imprimanta, daca iesirea curenta e la terminal, sau invers).

Forma generala este:

```
WRITE(p1,p2,...,pn);
```

echivalenta cu:

```
BEGIN WRITE (p1); WRITE(p2);...;WRITE(pn)
END;
```

unde argumentele p1, p2,..., pn pot fi de una din urmatoarele forme:

d sau d:i sau d:i:j sau d:i:H,

unde d este expresia a carei valoare se tipareste, i si j sint expresii intregi si H este o constanta.

-d este de tip INTEGER

WRITE(d)

se tipareste valoarea lui d, pe o lungime corespunzatoare numarului de cifre (la care se adauga eventual semnul), lasindu-se un spatiu in coada.

WRITE(d:i)

se tipareste valoarea lui d pe i pozitii; daca este cazul (i mai mare decit numarul de cifre) se adauga spatii in fata numarului.

```
    WRITE(d:i:H)
```

valoarea d se tipareste in forma hexazecimala; daca i=1 sau 2, se tiparesc i caractere; daca i=3 sau 4 se tipareste numarul complet hexazecimal pe patru caractere; daca i > 4 se adauga spatii in numar corespunzator in fata numarului hexaxecimal.

- d este de tip real

```
   WRITE(d)
   WRITE(d:i)
```

valoarea se tipareste in forma cu mantisa si exponent; daca i lipseste sau i < 8, numarul se tipareste pe 12 pozitii; daca 8 <= i<= 12, numarul se tipareste pe i pozitii, cu una sau mai multe (maxim 5) zecimale in mantisa. Daca i > 12 se adauga statii in fata numarului.

```
   WRITE(d:i:j)
```

numarul se tipareste in forma cu virgula fixa, cu j zecimale; i reprezinta dimensiunea totala a cimpului (numarul de pozitii); daca este cazul se aduga spatii in fata numarului; daca i este prea mic pentru numarul de zecimale specificat, se tipareste in forma cu mantisa si exponent, conform specificarii WRITE(d:i)

- d este de tip CHAR sau tablou (sir) de caractere

```
    WRITE(d)
```

se tipareste caracterul sau tabloul (sirul) de caractere;

```
WRITE(d:i)
```

se tipareste caracterul sau tabluol (sirul) de caractere pe i pozitii; daca este cazul, se competeaza cu spatii in fata.

- d este de tip booleelean

```
     WRITE(d)
```

se tipareste TRUE si FALSE, in functie de valoarea lui d;

```
    WRITE(d:i)
```

daca i > 4 respectiv 5, se adauga spatii in fata valorii tiparite.

Procedura standard WRITELN are ca efect incheierea liniei curente si trecerea la o linie noua.

```
    WRITELN(p1, p2,..., pn);
```

este echivalent cu

```
   BEGIN  WRITE(p1); WRITE(p2);...; WRITE(pn)
   WRITELN
   END;
```

Procedura standard PAGE provoaca stergerea ecranului, respectiv saltul la inceputul unei noi pagini la imprimanta.

Procedura standard READ se foloseste pentru introducerea unor date de la tastatura. Accesul la aceste date se realizeaza prin intermediul unui tampon, care initial este gol (mai precis contine un indicator de sfirsit linie - CHR(13)). Continutul acestui tampon este parcurs in ordine si la intilnirea indicatorului sfirsit linie se aduce o noua linie de la tastatura.

Forma generala este:

```
  READ(v1, v2,..., vn);
```

este echivalent cu:

```
   BEGIN READ(v1); READ(v2);...; READ(vn)
   END;
```

unde v1, v2,..., vn pot fi de tip caracter,

tablou de caractere, intreg sau real.

-v este de tip caracter;

urmatorul caracter din tamponul de la intrare este atribuit lui v; daca acest caracter este CHR(13), functia EOLN obtine valoarea TRUE, si se introduce o noua linie de la tastatura; la urmatorul READ se obtine primul caracter al acestei noi linii; ca o consecinta, daca primul READ din program citeste un caracter, acesta va fi intotdeaua CHR(13), si urmatoarea citire va trata primul caracter de pe linia introdusa; pentru a evita efectele neplacute ale acestui mod de lucru, se va folosi procedura READLN, prezentata mai jos.

-v este de tip tablou de caractere;

se citeste o succesiune de caractere, cite unul pentru fiecare element de tablou. Daca se ajunge la sfirsitl liniei (CHR(13)), restul tabloului se umple cu caracterul CHR(0). In continuare ramin valabile observatiile de mai sus privind citirea la inceput de program.

- v este de tip intreg;

se citeste o valoare intreaga; se ignora spatiile si indicatoarele de sfirsit linie care precede prima cifra (sau semn).

- v este de tip real;

se citeste o valoare in forma cu sau fara exponent, ca la valori intregi; se ignora spatiile si indicatoarele de sfirsit linie care preced prima linie (sau semn).

Procedura standard READLN.

Preia o noua linie de la tastatura; poate fi utilizata pentru eliminarea liniei vide prezente in tampon la lansarea in executie a programului, si introducerea primei linii de la tastatura; urmatorul READ va citi in acest caz primul caracter al liniei introduse.

```
READLN(v1, v2,...,vn);
```

este echivalent cu:

```
BEGIN READ(v1); READ(v2);...; READ(vn);
READLN END;
```

Functia standard EOLN.

Este de tip boolean si furnizeaza valoarea TRUE daca urmatorul caracter de citit este CHR(13); in caz contrar are valoarea FALSE.

Functia standard INCH.

Este de tip CHAR, baleiaza tastatura si daca gaseste o cheie apasata, returneaza valoarea corespunzatoare caracterului respectiv; in caz contrar furnizeaza valoarea CHR(0).

1.2.2.2. Functii aritmet ce si de conversie.

Functiile care se regasesc si in limbajul standard sint prezentate mai sumar.

- TRUNC(x conversie real-intreg, prin trunchiere
- ROUND(x) conversie real-intreg, prin rotunjire
- ENTIER(x) conversie real-intreg; returneaza cel mai mare numar intreg, mai mic decit x
  ENTIER(-4.3) returneaza -5
  ENTIER(8.8) returneaza 8
- ORD returneaza valoarea intreaga reprezentind numarul ordinal al valorii x de tip scalar.
- CHR(x) returneaza caracterul de ordinul x in tabelul ASCII
- ABS(x) returneaza valoarea absoluta a lui x
- SQR(x) returneaza valoarea x*x

- SQRT(x) returneaza radicalul lui x
- FRAC(x) returneaza partea fractionara a lui x
  FRAC(x)=x-ENTIER(x)
- SIN(x) sinus
- COS(X) cosinus
- TAN(x) tangenta
- ARCTAN(x) arctangenta
- EXP(x) returneaza valoarea lui e^x
- LN(x) logaritm natural

1.2.2.3. Alte proceduri si functii standard.

- NEW(p) - aloca spatii pentru o variabila dinamica; referinta la acesta variabila se returneaza prin variabila pointer p.
- MARK(v) - memoreaza starea zonei de alocare, in variabila pointer v.
- RELEASE(v)- eliberiaza zone de memorie alocate dinamic; reface starea zonei de alocare dinamica existenta in momentul in care s-a executat MARK(v).
- INLINE(c1, c2,..., cn)
  - permite inserarea in programul PASCAL a unor secvente in cod masina; valoarea constantelor intregi c1, c2,..., cn este introdusa in codul obiect, pe pozitia corespunzatoare din textul sursa.
- USER(x) - provoaca apelul unei proceduri la adresa de memorie x.
- HALT - opreste executia programului.
- POKE(x,v) - memoreaza valoarea expresiei v (de orice tip) in locatii succesive incepind cu cea de adresa x.
- TOUT(numel, adresa, dim)
  - permite memorarea unor valori pe caseta magnetica; fisierul care este asociat valorilor memorate, obtine numele transmis prin parametrul nume (sir de caractere); se salveaza un numar de octeti corespunzator valorii transmise prin dim, incepind cu octetul de la adresa transmisa prin cel de-al doilea parametru. TOUT('DATE', ADDR(v), SIZE(v)) memoreaza valoarea variabilei v intr-un fisier cu numerele DATE (a se vedea mai jos functiile standard ADDR si SIZE)
- TIN(nume, adresa)
  - permite incarcarea in memorie a unor valori de pe caseta magnetica; valorile se extrag din fisierul cu numele transmis prin parametrul nume (sir de caractere) si se memoreaza in locatii succesive, incepind cu cea de adresa transmisa prin cel de-al doilea parametru. TIN('DATA', ADDR(v)) incarca in fisirul numit DATE (creat anterior prin TOUT) in variabila v.
- OUT(p,c) - permite accesul direct la porturile de iesire ale microprocesorului Z80; valoarea intreaga transmisa prin p se incarca in registrul BC, caracterul transmis prin c se incarca in registrul A si se executa OUT(c), A.
- RANDOM - functie intreaga, care returneaza un numar aleator intre 0...255.
- SUCC(v) - functie care returneaza succesorul marimii scalare x.

- PRED(x) - functie care returneaza predecesorul marimii scalare x.
- ODD(x) - funtie booleana cu valoarea TRUE daca valoarea intreaga x este para si FALSE in caz contrar.
- ADDR(v) - functie intreaga care returneaza adresa variabilei v.
- SIZE(v) - functie intreaga care returneaza numarul de octeti alocati variabilei v.
- PEEK(x,t) - functie de tip t (al doilea parametru desemneaza tipul functiei); functia returneaza valoarea de la adresa de memorie x.

1.3. Comentarii si optiuni de compilare.

Comentariile se includ intre "{" si "}" sau intre "(*" si"*)". Caracterele intre cele doua semne se ignora, cu exceptia cazului in care primul caracter este "$". In acest caz semnul $ este urmat de o lista de optiuni de compilare, despartite intre ele prin caracterul ",".

Programatorul are la dispozitie urmatoarele optiuni de compilare mai importante:

- optiunea L, controleaza listarea programului sursa:
  - L+ programul se listeaza
  - L- se listeaza numai liniile in care a aparut o eroare.

  Implicit: L+.
- optiunea O, controleaza efectuarea unor verificari de depasire in calcule aritmetice;
  - O+ se verifica depasirea, la adunarea si scaderea intreaga
  - O- verificarile de mai sus nu se executa.

  Implicit: O+.
- optiunea A, controleaza efectuarea verificarilor de incadrare a indicilor de tablou intre limitele impuse
  - A+ se verifica incarcarea indicelui intre limite
  - A- nu se verifica.

  Implicit: A+
- optiunea P, controleaza dispozitivul la care se scoate listingul de compilare; la intilnirea optiunii se modifica dispozitivul la care se listeaza textul sursa, mai precis, daca pina atunci textul se afisa la terminal, in continuare se va lista la imprimanta si invers.

  Implicit: textul sursa se afiseaza la terminal.
- optiunea F, se va trata in paragraful 4.

2. Facilitati grafice pentru HISOFT-PASCAL

Impreuna cu compilatorul PASCAL, firma HISOFT furnizeaza si un pachet de programe, in format sursa, numit TURTLE. Acest pachet poate fi inclus de pe caseta in orice program PASCAL si folosit prin apelul procedurilor respective.

Ecranul este privit ca un cimp de 256x176 puncte, cu originea in coltul din stinga jos.

Prezentam mai jos cele mai importante facilitati puse la dispozitie prin procedurile pachetului TURTLE:

TURTLE -procedura de initializare, care trebuie apelata inainte de utilizarea rutinelor pachetului TURTLE;

ARCR(x,y) -procedura pentru trasarea unor arce; penita inainteaza de y ori cite x puncte, dupa care directia ei se roteste cu 1 grad;

LEFT(x) -directia de miscare a penitei se roteste cu x grade la stinga;

RIGHT(x) -directia de miscare a penitei se roteste cu x grade la dreapta;

BACK(x) -penita se intoarce cu x puncte;

VECTOR(x,y)
-directia de miscare a penitei se roteste cu x grade, dupa care inainteaza cu y puncte;

FWD (x) -penita inainteaza cu x puncte;

SET(x,y) -stabileste coordonate de plecare noi pentru penita;

PENUP -ridica penita; in aceasta stare, miscarile nu lasa urme;

PENDOWN(culoare)
-lasa penita in jos si stabileste culoarea conform conventiei:
culoare=0 -negru; 1 -albastru;
2 -rosu; 3 -violet; 4 -verde;
5 -bleu; 6 -galben; 7 -alb;

COPY -copiaza ecranul la imprimanta;

PAPER(culoare) -stabileste culoarea ecranului conform convenientei de la PENDOWN;

INK(culoare) -stabileste culoarea penitei conform conventiei de la PENDOWN;

LINE(x,y) -traseaza o linie intre punctul curent S(X,Y) si punctul P(X+x,Y+y)

PLOT(x,y) -deseneaza punctul P(x,y).

3. Editarea programului in HISSOFT-PASCAL

Sistemului HISOFT-PASCAL ii este atasat un editor de texte, incarcat in mod automat, impreuna cu compilatorul si executivul. In continuare se prezinta comenzile care permit introducerea si modificarea unui program. Acestea se introduc ca raspuns la promterul sistemului ">".

In,m -trece sistemul in regim de introducere linii; liniile se numeroteaza incepind cu n, cu pasul m; iesirea din regimul de introducere se face tastind CAPS SHIFT si 1;

Dn,m -se sterg liniile intre cele de numar n si m (inclusiv limitele); daca m lipseste sau are aceeasi valoare cu n, se sterge linia cu numarul n;

Fn,m,f,s - se cauta in liniile incepind de la cea de numar n pina la m, secventa de caractere f; prin apasarea tastei S, se poate cere inlocuirea acestei secvente cu sirul s; daca nu se doreste inlocuirea, se tasteaza F;

Mn,m -se copiaza linia n in linia m;

Nn,m -se renumeroteaza textul programului, incepind cu linia n si pasul m;

Ln,m -daca n si m lipsesc, se listeaza toate liniile; n si m reprezinta numarul primei, respectiv a ultimei linii de listat;

En -urmeaza sa fie editata linia n; in acest scop se folosesc urmatoarele comenzi:
SPACE
muta cursorul in cadrul liniei la dreapta
DELETE
muta cursorul in cadrul liniei la stinga

ENTER
incheie editarea liniei si o memoreaza in noua forma
Q
incheie editarea liniei si o memoreaza in vechea forma
K
sterge caracterul indicat de cursor
I
insereaza caracterul pe pozitia indicata de cursor; dupa ultimul caracter se tasteaza ENTER;
X
salt la sfirsitul liniei
C
inlocuirea caracterului indicat de cursor prin caracterul tastat; dupa ultima modificare se tasteaza ENTER.

4. Comenzi de lucru cu caseta magnetica. Alte comenzi sistem.

Comenzile de mai jos se introduc ca raspuns la promterul sistem ">".

Cn -se compileaza textul incepind cu linia n; daca n lipseste, se compileaza incepind cu prima linie;

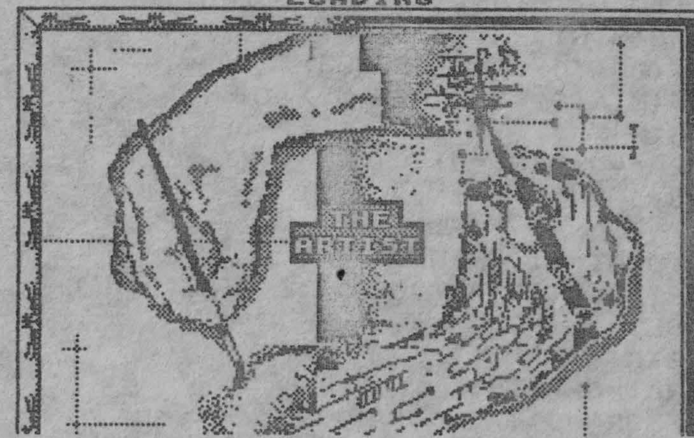
B -se paraseste sistemul PASCAL si se revine in BASIC; trecerea din BASIC in PASCAL se realizeaza prin comanda RANDOMIZE USR 24603;

Pn,m,s -se salveaza textul, intre liniile n si m pe caseta, intr-un fisier caruia i se asociza numele s;

G,,s -se incarca un program de pe caseta (program salvat anterior printr-o comanda P); textul incarcat este cel din fisierul cu numele s; daca s lipseste din comanda, se incarca urmatorul program de pe caseta; daca exista un text deja introdus, atunci programul incarcat se ataseaza in continuarea celui existent;

Wn,m,s -are acelasi efect ca si P, cu deosebirea ca textul salvat se poate incarca in interiorul unui program, prin optiunea de

LOADING



incarcare F:
{$Fs} - la compilare se incarca pe locul respectiv, din fisierul s, un text salvat anterior cu W.

Tn -are ca efect compilarea programului, incepind cu linia n (daca n lipseste, se compileaza incepind cu prima linie), salvarea codului obiect (impreuna cu rutinele executivului) executabil pe caseta magnetica. Incarcarea si lansarea in executie a programului se face prin comanda LOAD din BASIC.

5. Aplicatii in limbajul HISOFT-PASCAL

Programul EXIF realizeaza transformarea orei exprimata intre 0 si 24 in ora exprimata de la 0 la 12 AM si PM.

```
PROGRAM EXIF;
VAR
 ORA,MIN:INTEGER;
BEGIN
 WRITELN('ORA?');
 READ(ORA);
 WRITELN('MIN');
 READ(MIN);
 IF (ORA<0) OR (ORA>23) OR
    (MIN<0) OR (MIN>59) THEN
 BEGIN
  ORA:=0;MIN:=0;
 END;

 IF ORA<=11 THEN
  WRITELN('ORA:',ORA,':',MIN,' AM')
 ELSE
 BEGIN
  ORA:=ORA-12;
  IF ORA=0 THEN ORA:=12;
  WRITELN('ORA:',ORA,':',MIN,' PM');
  END;
END.
```

Programul EXWHILE tipareste puterile lui 2 si media aritmetica a unui sir de numere reale pozitive citite de la tastatura. Programul se termina la introducerea primului numar negativ.

```
PROGRAM EXWHILE;
VAR
   N,P:INTEGER;
   INNUM,SUM:REAL;
BEGIN
 N:=0;P:=1;
 WHILE P<MAXINT DIV 2 DO
  BEGIN
   WRITELN(N:2,P:4);N:=N+1;
   P:=P*2;
  END;
  WRITELN(N:2,P:4);
  SUM:=0;N:=0;
  WRITELN('INTRODUCETI NR.>0');READ(INNUM);
  WHILE INNUM>=0 DO
   BEGIN
    SUM:=SUM+INNUM;
    N:=N+1;
    WRITELN('   ',INNUM);
    READ(INNUM);
   END;
  IF N>0 THEN WRITELN('MEDIA ARITM:',SUM/N:5:2);
 END.
```

Programul EXREPEAT tipareste puterile lui 2 si citeste de la tastatura o secventa de numere intregi, terminindu-se la introducerea unui 0.

```
PROGRAM EXREPEAT;
VAR
  N,P:INTEGER;
  INCHAR:CHAR;
BEGIN
 N:=0;P:=1;
 REPEAT
  N:=N+1;P:=P*2;
  WRITELN(N:3,' ',P:6);
 UNTIL P>MAXINT DIV 2;
 REPEAT
 READ(N);WRITE(':',N);WRITELN;
 UNTIL N = 0;
END.
```

Programul EXFOR calculeaza patratele numerelor naturale si puterile lui 2. De asemenea se citesc de la tastatura 2 caractere si apoi se tiparesc toate caracterele cuprinse intre ele.

```
PROGRAM EXFOR;
VAR
 I,P,N,M,NSUM:INTEGER;
 CHAR1,CHAR2,PCHAR,FICTIV:CHAR;
 INNUM,SUM:REAL;
```

```
BEGIN
 FOR I:=1 TO 10 DO
  WRITELN(I:2,I*I:4);
  P:=1;
  FOR I:=1 TO 10 DO
   BEGIN
   WRITELN(I:2,P:4);P:=P*2;
   END;
  WRITELN('INTRODUCETI 2 CARACTERE SI <ENTER>');
 READ(FICTIV);
 READ(CHAR1,CHAR2);
 IF CHAR1<CHAR2 THEN
  FOR PCHAR:=CHAR1 TO CHAR2 DO WRITELN(PCHAR)
 ELSE
  FOR PCHAR:=CHAR1 DOWNTO CHAR2 DO WRITELN(PCHAR);
END.
```

Programul EXCASE citeste de la tastatura cite un caracter si recunoaste daca acesta este o litera corespunzatoare unei vocale sau o cifra. Se termina prin introducerea unui ".".

```
    PROGRAM EXCASE;
VAR V:CHAR;
BEGIN
 READ(V);
 WHILE V <> '.' DO
  BEGIN
   CASE V OF
    'A','E','I','O','U':WRITELN('VOCALA');
     '0','1','2','3','4','5','6','7','8','9':WRITELN('CIFRA')
   END;
   READ(V);
  END;
END.
```

Programul EXSET reprezinta o aplicatie a tipului multime. Se citeste de la tastatura un sir de caractere care se grupeaza cite 4. Programul verifica daca intr-un grup dat, toate caracterele reprezinta cifre pare sau impare. Atunci cind grupul se incadreaza in una din aceste situatii se tipareste un mesaj corespunzator, suma cifrelor si cifrele respective. Daca intre cele 4 cifre una se repeta de mai multe ori, ea se tipareste doar o singura data.

```
PROGRAM EXSET;
TYPE BAZA=0..9;
VAR
 MAN,MULT,IMP,PAR:
   SET OF 0..9;
 NUMAR: SET OF CHAR;
 C:CHAR;
 I,SUM:INTEGER;
 CONTOR:INTEGER;
 SF:BOOLEAN;

PROCEDURE CARNOU;
 BEGIN
  REPEAT
    BEGIN
   READ(C);
{ IF EOLN THEN BEGIN WRITELN(C,' ',ORD(C)) END;I}
    END;
  UNTIL (C IN NUMAR) OR           (C='%');
  SF:=C='%';
 END;
```



```
BEGIN
 WRITELN('INTRODUCETI UN SIR DE CARACTETR TERMINAT CU %');  READLN;
 IMP:=[1,3,5,7,9];
 PAR:=[0,2,4,6,8];
 NUMAR:=['0','1','2','3','4','5','6','7','8','9'];
 MULT:=[];
 CONTOR:=0;
 CARNOU;
 WHILE NOT SF DO
  BEGIN
   SUM:=0;
   CONTOR:=CONTOR+1;
   FOR I:=1 TO 4 DO
   BEGIN
    IF NOT SF THEN
     BEGIN
      MAN:=[ORD(C)-                    ORD('0')];
      MULT:=MULT + MAN;
      SUM:=SUM+ORD(C)-                  ORD('0');
      CARNOU;
     END;
   END;
 IF MULT<=IMP THEN
   BEGIN
    WRITELN;
    WRITELN('GR:',CONTOR,    ' IMPAR,SUMA=',SUM);
    I:=1;
    REPEAT
     BEGIN
      IF I IN MULT THEN
       WRITE(I,' ');
      I:=I+2;
     END;
    UNTIL I>9;
    WRITELN;
    END
  ELSE
   BEGIN
   IF MULT<=PAR THEN
```

i$M_4




```
BEGIN
 WRITELN;
 WRITELN('GR:',CONTOR       ,'PAR,SUMA=',SUM);
 I:=0;
 REPEAT
  BEGIN
   IF I IN MULT THEN
    WRITE(I,' ');
   I:=I+2;
  END;
 UNTIL I>8;
END;
WRITELN;
END;
MULT:=[];
END;
END.
```

Programul EXARRAY calculeaza pentru patru examene in cazul a cel mult 20 de studenti, media studentului, media pe examen si media pe examen si student. Notele sint pastrate in tabloul NOTA si mediile pe student in tabloul MSTUD. Mediile pe examene si media generala se pastreaza in variabile simple.

```
PROGRAM EXARRAY;
CONST
 MAXSTUD=20;
 NEXAM=4;
TYPE
    S=1..MAXSTUD;
    E=1..NEXAM;
VAR
 IS,NRSTUD:S;
 IE:E;
 NOTA:ARRAY[S,E] OF REAL;
 MSTUD:ARRAY[S] OF REAL;
 SUM1,SUM2,MEDIA:REAL;
BEGIN
  WRITELN('NR. STUD.=?');
 READ(NRSTUD);
 SUM1:=0;
 FOR IS:=1 TO NRSTUD DO
  BEGIN
   MSTUD[IS]:=0;
   WRITELN('NOTE PT. STUD. ',IS);
  FOR IE:=1 TO NEXAM DO
   BEGIN
   READ(NOTA[IS,IE]);
   SUM1:=SUM1+NOTA[IS,IE];
   MSTUD[IS]:=MSTUD[IS]+NOTA[IS,IE];
   END;
  WRITELN('MEDIA STUD ',IS,' ',MSTUD[IS]/NEXAM);
   WRITELN(' ');
  END;
  FOR IE:=1 TO NEXAM DO
   BEGIN
   SUM2:=0;
   FOR IS:=1 TO NRSTUD DO
    SUM2:=SUM2+NOTA[IS,IE];
   WRITELN('MEDIA EXAM ',IE,' ',SUM2/NRSTUD);
   END;
  WRITELN('MEDIA GENERALA:',SUM1/(NEXAM*NRSTUD));
END.
```

Programele EXP1, EXP2, EXP3 si EXP4 exemplifica posibilitatile de transfer a parametrilor constanti si varabili spre procedura S (care calculeaza A=B+C) si notiunea de domeniu de vizibilitate a unui identificator. In EXP1 si EXP2 variabilele A, B, C sint globale, deci lista parametrilor formali poate fi vida (definirea ei in EXP2 nefiind necesara). In EXP3 variabilele A, B, C sint globale si de asemenea locale procedurii S. In acest caz rezultatul este incorect intrucit B si C fiind parametrii formali constanti (de intrare) nu au valoarea dorita in procedura. In EXP4, B si C sint de asemenea variabile locase procedurii S, carora li se atribue valoarea parametilor formali Y si Z calculindu-se corect suma A. Aceasta in final se atribuie lui Z care este un parametru variabil (de iesire) al procedurii S.

```
         PROGRAM EXP1;
VAR A,B,C:INTEGER;
PROCEDURE S;
BEGIN
WRITELN('IN PROCEDURA');
WRITELN('A=',A,'B=',B,'C=',C);
A:=B+C;
WRITELN('A=',A);
END;

BEGIN
A:=0;
C:=1;B:=2;
```

```
    WRITELN('P.  R.INAINTE DE APEL PROCEDURA:');
    WRITELN('A=',A,'B=',B,'C=',C);
    S;WRITELN('P. PR. DUPA APEL PROCEDURA ');WRITELN('A=',A);
    END.

PROGRAM EXP2;
VAR A,B,C:INTEGER;
PROCEDURE S(VAR A:INTEGER);
BEGIN
WRITELN('IN PROCEDURA');
WRITELN('A=',A,'B=',B,'C=',C);
A:=B+C;
WRITELN('A=',A);
END;

BEGIN
A:=0;
C:=1;B:=2;
WRITELN('P. PR.INAINTE DE APEL PROCEDURA:');
WRITELN('A=',A,'B=',B,'C=',C);
S(A);WRITELN('P. PR. DUPA APEL PROCEDURA ');WRITELN('A=',A);
END.
PROGRAM EXP3;
VAR A,B,C:INTEGER;
PROCEDURE S(VAR A:INTEGER;B,C:INTEGER);
VAR A,B,C:INTEGER;

BEGIN
WRITELN('IN PROCEDURA');
WRITELN('A=',A,'B=',B,'C=',C);
A:=B+C;
WRITELN('A=',A);
END;

BEGIN
A:=0;
C:=1;B:=2;
WRITELN('P. PR.INAINTE DE APEL PROCEDURA:');
WRITELN('A=',A,'B=',B,'C=',C);
S(A,B,C);
WRITELN('P. PR. DUPA APEL PROCEDURA ');WRITELN('A=',A);
END.


PROGRAM EXP4;
VAR A,B,C:INTEGER;
PROCEDURE S(VAR X:INTEGER;Y,Z:INTEGER);
VAR A,B,C:INTEGER;
BEGIN
A:=X;B:=Y;C:=Z;
WRITELN('IN PROCEDURA');
WRITELN('A=',A,'B=',B,'C=',C);
A:=B+C;
WRITELN('A=',A);
X:=A;
END;
```

```
BEGIN
A:=0;
C:=1;B:=2;
WRITELN('P. PR.INAINTE DE APEL PROCEDURA:');
WRITELN('A=',A,'B=',B,'C=',C);
S(A,B,C);
WRITELN('P. PR. DUPA APEL PROCEDURA ');WRITELN('A=',A)
END.
```

Programul EXSORT prezinta cautarea unui anumit element intr-un tablou prin metoda cautarii binare aplicata unui sir de elemente ordonate. Pentru aceasta se sorteaza crescator elementele tabloului A cu 5 elemente, care in stare initala contin valorile: ART5, ART4, ..., ART1. Pentru sortare se foloseste algoritmul lui Hoare, numit Quicksort. Aceasta sortare, prin partitionare porneste de la urmatorul algoritm. Fie X un element oarecare al tabloulu de sortat A1.....An. Se parcurge tabloul de la stinga pina se gaseste primul element Ai>X. Acum se parcurge tablou de la dreapta pina se gaseste primul element Aj<X. Se schimba intre ele elementele Ai si Aj, apoi se continua parcurgerea tabloului de la stinga si de la dreapta (din punctele in care s-a ajuns anterior) pina se gasesc alte 2 elemente care se schimba intre ele s.a.m.d.. Procesul se termina cind cele 2 parcurgeri "se intilnesc" undeva in interiorul tabloului. Efectul final este ca acum sirul intial este partitionat intr-o parte stinga cu chei < X si o parte dreapta cu chei > X. Dupa o prima partitionare a secventei de elemente se aplica aceeasi procedura celor doua partitii rezultate, apoi celor patru partitii ale acestora, s.a.m.d. pina cind fiecare partitie se reduce la un singur element. Procedura SORTARE se apeleaza recursiv pe ea insasi. In urma sortarii tabloul A contine informatiile: ART1, ART2, ..., ART5. Mai departe programul va determina pe ce pozitie in tabloul A se gaseste elementul ce contine informatia "ART3".

In cazul unui tablou sortat, principiul de cautare cel mai des folosit este metoda injumatatirii intervalui numita si cautarea binara. La fiecare repetare intervalul inspectat situat intre indicii i si j este injumatatit.

```
PROGRAM EXSORT;
CONST
 N=5;
 ART='ART';
TYPE
 INDICE=0..N;
 ELEMENT=RECORD
  CIMP:ARRAY[1..3] OF CHAR;
  CHEIE:INTEGER;
 END;
VAR
 I:INTEGER;
 A:ARRAY[1..N] OF ELEMENT;
 X:ELEMENT;
 M,NN,K:INDICE;

PROCEDURE SORTARE(VAR S,D:INDICE);

VAR
  I,J:INDICE;
  X,W:ELEMENT;
BEGIN
 I:=S;J:=D;
 X:=A[(S+D) DIV 2];
 REPEAT
  WHILE A[I].CHEIE<X.CHEIE  DO I:=I+1;
  WHILE X.CHEIE<A[J].CHEIE DO J:=J-1;
  IF I<=J THEN
   BEGIN
    W:=A[I];A[I]:=A[J];A[J]:=W;
```

```
   I:=I+1;J:=J-1
  END
 UNTIL I>J;
 IF S<J THEN SORTARE(S,J);
 IF I<D THEN SORTARE(I,D);
END;
BEGIN
 WRITELN('TABLOUL NESORTAT');
 FOR I:=1 TO N DO
  BEGIN
  A[I].CIMP:=ART;
  A[I].CHEIE:=N+1-I;WRITE(A[I].CIMP,A[I].CHEIE,' ');
  END;
 M:=1;NN:=N;
 SORTARE(M,NN);
WRITELN;WRITELN; WRITELN('TABLOUL SORTAT');
 FOR I:=1 TO N DO
  WRITE(A[I].CIMP,A[I].CHEIE,' ');
WRITELN;WRITELN; WRITELN('SE CAUTA EL.=ART3');
 M:=1;NN:=N;X.CIMP:=ART;X.CHEIE:=3;
 REPEAT
  BEGIN
  K:=(M+NN) DIV 2;
  IF X.CHEIE>A[K].CHEIE THEN M:=K+1 ELSE NN:=K-1
  END
  UNTIL (A[K].CHEIE=X.CHEIE) OR (M>NN);
 IF M<=NN THEN WRITELN('ART. CAUTAT E PE POZ. ',K)
END.
```

In programul EXHANOI se prezinta rezolvarea problemei cunoscuta sub numele de "Turnurile din Hanoi" avind urmatorul enunt:

-Se dau trei bare identice de lemn fixate pe o masa si un numar oarecare de discuri de lemn avind un orificiu la mijloc astfel incit sa poata fi introduse pe una din bare. Discurile au diametre diferite ,neexistind doua la fel.Stiind ca discurile sint asezate pe prima bara (stinga) in ordinea marimilor de jos in sus se cere, sa se transfere toate discurile pe ultima bara (dreapta), in final avind aceeasi ordine ca in starea initiala. Rezolvarea problemei se face stiind ca la un moment dat se poate muta un singur disc de pe o bara pe alta, mutare in urma careia pe fiecare bara discurile trebuie sa fie asezate astfel incit sa se respecte restrictia initiala cu privire la ordonarea in functie de marimea lor.

Algoritmul pentru mutarea celor n discuri este prezentat schematic in fig. 1 In prima operatie se transfera (n-1) discuri pe bara din mijloc. A doua operatie muta discul n de pe bara din stinga pe bara din dreapta, iar operatia a treia realizeaza mutarea celor (n-1) discuri din mijloc din bara din dreapta. Dupa cum se observa operatia "a doua" si "a treia" sint de fapt secvente de operatii.

Programul implementeaza in mod recursiv acest algoritm. In programul principal se cere prin dialog numarul de discuri n si se apeleaza procedura recursva MOVE, transmitindu-i-se starea initiala:

-numarul de discuri si

-identitatea din starea initiala a barelor

Procedura MOVE (N:INTEGER, SCE, AUX, DEST:B) verifica daca N=1 situatie in care se mut discul si algoritmul este incheiat. In caz contrar se apeleaza procedura MOVE pentru (n-1) discuri si noua identitate a barelor, pentru a face mutarea de la stinga la mijoc a celor (n-1) discuri. La revenirea din procedura MOVE se apeleaza procedura MOVEDISK, pentru a muta discul n de pe bara stinga pe bara dreapta. Mai departe se apeleaza din nou MOVE cu parametrii corespunzatori realizarii mutarii celor (n-1) discuri de pe bara din mijloc pe bara din dreapta.

Procedura MOVEDISK realizeaza transferul discului din stinga (SCE) pe bara din dreapta

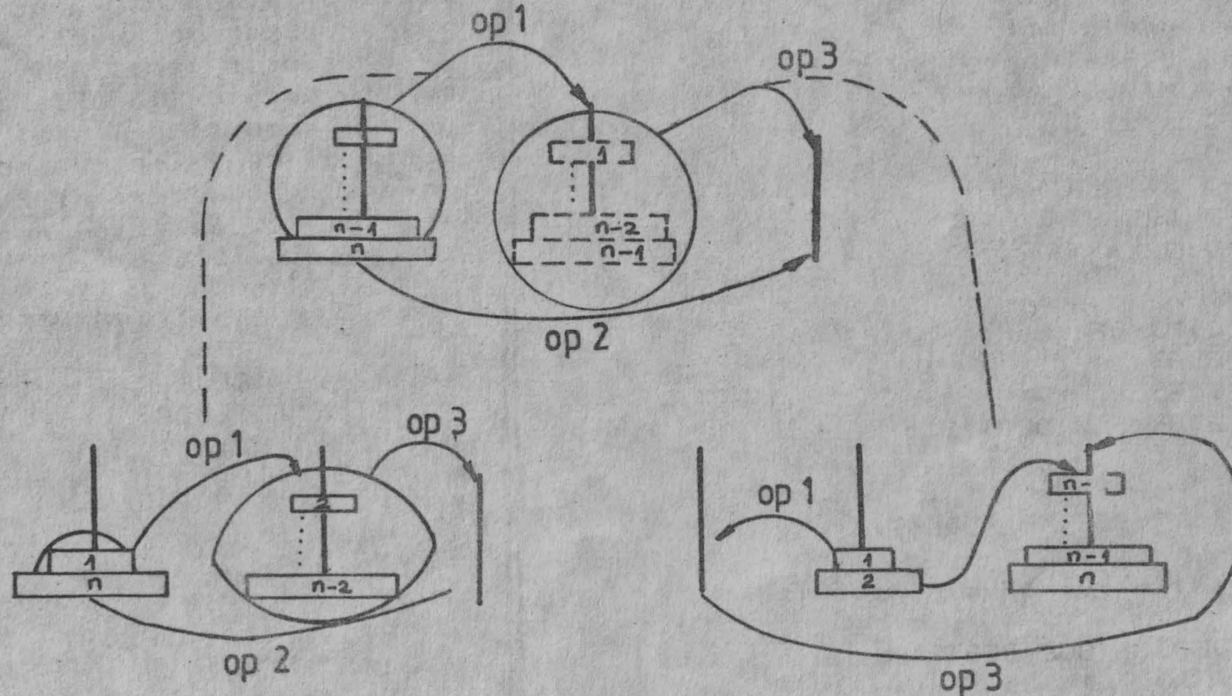


Fig. 1.

(DEST), tiparind mesajul respectiv.

Secventa de solutii a problemei este generata in momentul revenirii din apelul procedurilor recursive.

Rulind programul pentru N=3 pe ecran se obtine urmatoarea solutie:

```
   1: STINGA  -> DREAPTA
   2: STINGA  -> MIJLOC
   3: DREAPTA -> MIJLOC
   4: STINGA  -> DREAPTA
   5: MIJLOC  -> STINGA
   6: MIJLOC  -> DREAPTA
   7: STINGA  -> DREAPTA
```

```
PROGRAM HANOI;
TYPE B=ARRAY[1..7] OF CHAR;
VAR N,NUM:INTEGER;
   SCE,AUX,DEST:B;
PROCEDURE MOVEDISK(SCE,DEST:B);
BEGIN
NUM:=NUM+1;
WRITELN(NUM:4,': ',SCE,'->',DEST);
END;
PROCEDURE MOVE(N:INTEGER;SCE,AUX,DEST:B);
BEGIN
IF N=1 THEN MOVEDISK(SCE,DEST)
ELSE
BEGIN
MOVE(N-1,SCE,DEST,AUX);
MOVEDISK(SCE,DEST);
MOVE(N-1,AUX,SCE,DEST);
END;
END;
BEGIN
WRITE('CITE DISCURI SINT?');
READ(N);WRITELN;
NUM:=0;
MOVE(N,'STINGA ','MIJLOC ','DREAPTA');
END.
```

In programul EXPOINTER se construieste o structura de date recursiva. Procedura INIT genereaza dinamic prin procedura standard NEW variabila VP^, care reprezinta in final o lista liniara avind un numar de noduri ce depinde de valoarea parametrului N. Un nod al listei este descris prin tipul ART. Componentele LEG si LEG1 de tip pointer la ART, asigura inlantuirea ordonata a nodurilor listei si posibilitatea parcurgerii ei de la primul nod creat spre ultimul si invers. Procedura TIP tipareste lista creata, accesul la ea fiind asigurat prin variabila de tip pointer VPL.

```
     PROGRAM EXPOINTER;
TYPE
 ART=RECORD
  CIMP:INTEGER;
  LEG:^ART;
  LEG1:^ART;
  END;
  PT=^ART;
 IND=1..10;
VAR
 VPI,VPF,VPD,VPL:PT;
 I:INTEGER;
 SENS:ARRAY[1..2] OF CHAR;

PROCEDURE INIT(N:IND);
 BEGIN
  NEW(VPI);VPD:=VPI;
  VPF:=NIL;
  FOR I:=N TO N+3 DO
   BEGIN
    VPI^.CIMP:=I;
    VPI^.LEG:=VPF;
    VPF:=VPI;
    NEW(VPI);
    VPF^.LEG1:=VPI;
```

```
      END;
      VPF^.LEG1:=NIL;
    END;

   PROCEDURE TIP;
    BEGIN
    WRITELN;
     WHILE VPL<>NIL DO
      BEGIN
      WRITE(VPL^.CIMP);
      IF SENS='DR' THEN
       VPL:=VPL^.LEG
       ELSE
       IF SENS='ST'
        THEN
        VPL:=VPL^.LEG1;
      END;
    END;

   BEGIN
    INIT(2);
    VPL:=VPF;SENS:='DR'   ;
    TIP; INIT (1);
    VPL:=VPD;SENS:='ST'    ;
    TIP;
   END.
```

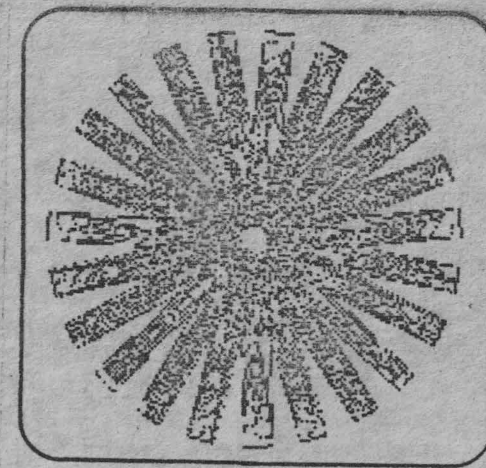
Programul EXMARK ilustreaza posibilitatea alocarii si eliberarii dinamice a zonelor de memorie. Prin procedura standard MARK se indica in variabila de tip pointer VPM adresa la care se readuce stiva variabilelor dinamice in momentul in care se va executa eliberarea, prin procedura standard RELEASE, a zonelor alocate dinamic pina in acel moment prin procedura standard NEW. In urma executarii unei eliberari a zonelor de memorie alocate dinamic noile alocari dinamice prin NEW vor crea variabile incepind cu adresele marcate initial prin executarea lui MARK(VPM). Rulind programul se observa ca in urma executarii RELEASE-ului noile variabile VP2^ vor fi create incepind cu adresa de la care sa creat initial variabila VP1^, adresa care a fost retinuta la inceputul programului prin MARK(VPM).

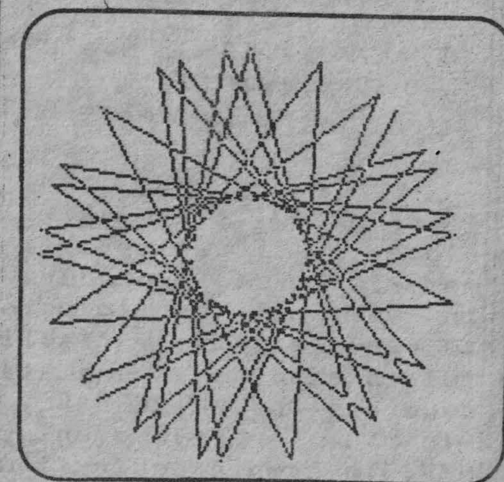
```
PROGRAM EXMARK;
TYPE
    EX= INTEGER ;
VAR VP1,VP2,VPM:^EX;
BEGIN
    MARK(VPM);{!!}
 NEW(VP1);VP1^:=1;
 NEW(VP2);VP2^:=2;
 WRITELN(VP1^,VP2^);
 VP2^:=VP1^+100;
 WRITELN(VP2^);
 WRITELN(ADDR(VP1^),ADDR(VP1),PEEK(ADDR(VP1^),INTEGER));
 WRITELN(ADDR(VP2^),ADDR(VP2),PEEK(ADDR(VP2^),INTEGER));
 NEW(VP2);VP2^:=3;
  WRITELN(ADDR(VP2^),ADDR(VP2) PEEK(ADDR(VP2^),INTEGER));
      RELEASE(VPM);{!!}
 NEW(VP2);VP2^:=4;
  WRITELN(ADDR(VP2^),ADDR(VP2),PEEK(ADDR(VP2^),INTEGER));
 NEW(VP2);VP2^:=5;
  WRITELN(ADDR(VP2^),ADDR(VP2),PEEK(ADDR(VP2^),INTEGER));
 NEW(VP2);VP2^:=6;
  WRITELN(ADDR(VP2^),ADDR(VP2),PEEK(ADDR(VP2^),INTEGER));
END.
```

Programul GRAFIC este o aplicatie a procedurilor pentru prelucrari grafice din pachetul de programe TURTLE furnizat de firma HISOFT.

```
    PROGRAM GRAFIC;
VAR I:INTEGER;
XCOR,YCOR,HEADING:REAL;
PENSTATUS:INTEGER;
PROCEDURE SPOUT(C:CHAR);
BEGIN
 INLINE(#FD,#21,#3A,#5C,#DD,#7E,2,#D7)
```

```
END;
PROCEDURE CHECK(X,Y:INTEGER);
BEGIN
 IF (X>255) OR (X<0) OR (Y>175) OR (Y<0) THEN
 BEGIN
  WRITE('Out of limits');
  HALT
 END;
 XCOR:=X;YCOR:=Y;
END;
PROCEDURE PLOT(X,Y:INTEGER);
BEGIN
 CHECK(X,Y);
 SPOUT(CHR(20));SPOUT(CHR(PENSTATUS));SPOUT(CHR(21));SPOUT(CHR(PENSTATUS));
 INLINE(#FD,#21,#3A,#5C,#DD,#46,2,#DD,#4E,4,#CD,#E5,#22)
END;
PROCEDURE LINE1(X,Y,SX,SY:INTEGER);
BEGIN
 INLINE(#FD,#21,#3A,#5C,#DD,#56,2,#DD,#5E,4,#DD,#46,6,#DD,#4E,8,#CD,#BA,#24)
END;
PROCEDURE LINE(ON:BOOLEAN;X,Y:INTEGER);
VAR SGNX,SGNY:INTEGER;
BEGIN
 CHECK(ROUND(X+XCOR),ROUND(Y+YCOR));
 SPOUT(CHR(20));SPOUT(CHR(PENSTATUS));SPOUT(CHR(21));SPOUT(CHR(PENSTATUS));
 IF X<0 THEN SGNX:=-1 ELSE SGNX:=1;
 IF Y<0 THEN SGNY:=-1 ELSE SGNY:=1;
 LINE1(ABS(X),ABS(Y),SGNX,SGNY)
END;
PROCEDURE INK(C:INTEGER);
BEGIN
 IF (C>=0) AND (C<8) THEN
 SPOUT(CHR(16));SPOUT(CHR(C))
END;
PROCEDURE PAPER(C:INTEGER);
BEGIN
 IF (C>=0) AND (C<8) THEN
 INLINE(1,0,3,#21,0,#58,#DD,#7E,2,7,7,7,#5F,#7E,#E6,#C7,#B3,#77,#23,#0B,#78,
```





```
#B1,#20,#EE);
 SPOUT(CHR(17));SPOUT(CHR(8));
END;
PROCEDURE COPY;
BEGIN
 INLINE(#FD,#21,#3A,#5C,
        #FD,#CB,#01,#CE,
        #CD,#AC,#0E,#FD,
    #CB,#01,#8E,#F3,#C9)

END;
PROCEDURE PENDOWN(C:INTEGER);
BEGIN
 PENSTATUS:=0;
 INK(C)
END;
PROCEDURE PENUP;
BEGIN
 PENSTATUS:=1
END;
PROCEDURE SETHD(A:REAL);
BEGIN
 HEADING:=A
END;
PROCEDURE SETXY(X,Y:REAL);
BEGIN
 XCOR:=X;
 YCOR:=Y
END;
PROCEDURE FWD(L:REAL);
VAR NEWX,NEWY:REAL;
BEGIN
 PLOT(ROUND(XCOR),ROUND(YCOR));
 NEWX:=XCOR+L*COS(HEADING*3.1415926/180);
 NEWY:=YCOR+L*SIN(HEADING*3.1415926/180);
 LINE(TRUE,ROUND(NEWX)-ROUND(XCOR),ROUND(NEWY)-ROUND(YCOR));
 XCOR:=NEWX;
 YCOR:=NEWY
END;
```

```
PROCEDURE TURN(A:REAL);
BEGIN
 HEADING:=HEADING+A
END;
PROCEDURE TURTLE;
BEGIN
 PAGE;
 SETXY(127,87);
 SETHD(0);
 PAPER(1);
 PENDOWN(6)
END;
PROCEDURE BACK(L:REAL);
BEGIN
 FWD(-L)
END;
PROCEDURE VECTOR(A,L:REAL);
BEGIN
 SETHD(A);
 FWD(L)
END;
PROCEDURE RIGHT(A:REAL);
BEGIN
 TURN(-A)
END;

PROCEDURE LEFT(A:REAL);
BEGIN
 TURN(A)
END;
PROCEDURE ARCR(R:REAL;A:INTEGER);
VAR I:INTEGER;
BEGIN
 FOR I:=1 TO A DO
 BEGIN
  FWD(R);TURN(1)
 END
END;
PROCEDURE SEITE(L,W,Z,MAX:INTEGER);
```

```
BEGIN
 IF L<MAX THEN BEGIN
 FWD(L);
 TURN(W);
 L:=L+Z;
 SEITE(L,W,Z,MAX)
 END
END;
PROCEDURE FIG(G,X,Y,L:INTEGER);
VAR I:INTEGER;
BEGIN
 IF G=1 THEN BEGIN
 SETXY(X,Y);
 SETHD(90);
 FOR I:=1 TO 5 DO BEGIN
  FWD(L);
  SETXY(X,Y);
  TURN(72)
 END
END ELSE BEGIN
 SETHD(0);
 FOR I:=1 TO 5 DO BEGIN
  FWD(L);
  FIG(G-1,TRUNC(XCOR),TRUNC(YCOR),L DIV 2);
  SETXY(X,Y);
  SETHD(I*72)
 END;
END
END;
PROCEDURE ROTPOL;
VAR L,W,Z,MAX:INTEGER;
BEGIN
 WRITE('SEG INITIAL= ');
 READ(L);
 WRITE('UNGHIUL= ');
 READ(W);
 WRITE('INC SEG= ');
 READ(Z);
 WRITE('MAX SEG= ');
 READ(MAX);
 PAGE;
 TURTLE;
 SEITE(L,W,Z,MAX)
END;
PROCEDURE FULGI;
VAR ORDN,X,Y,DIM:INTEGER;
BEGIN
 WRITELN('FULGI');
 X:=127;Y:=87;
 WRITE('ORDIN= ');
 READ(ORDN);
 WRITE('LUNGIME= ');
 READ(DIM);
 TURTLE;
 FIG(ORDN,X,Y,DIM);
END;
BEGIN
 WRITELN('1=ROTPOL');
 WRITELN('2=FULGI');
 WRITE('I= ');
 READ(I);
 IF I=1 THEN ROTPOL
  ELSE
   IF I=2 THEN FULGI
    ELSE WRITE('EROARE')
END.
```


Bibliografie:

L.D.Serbaniti, V.Cristea, F.Moldoveanu, V.Iorga: Programarea sistemelor in limbajele PASCAL si FORTRAN, Ed. Tehnica, Bucuresti 1984.

Vl. Cretu: Structuri de date si tehnici de programare, Lit. I.P.T. 1986

H.Ciocirlie, P. Eles, I. Balla: Limbajele de programare PASCAL si PASCAL concurent, Ed. Facla, Timisoara 1985.

1 COMENZI DE EDITARE

* I n,m * -inserare de text sursa
n - numarul liniei
m - pasul de numerotare
-dupa fiecare linie se tasteaza <enter>, moment in care apare numarul liniei urmatoare
-iesirea din inserare se face cu <caps-1>

* L n,m * -listare program
n-numarul primei linii listate
m-numarul ultimei linii listate

* D n,m * -stergere linii
n-numarul primei linii sterse
m-numarul ultimei linii sterse

* N n,m * -renumerotare a programului
n-noua etichetei a primei linii
m-noul pas de numerotare

* E n * -editarea liniei n

-Caractere de control ale editorului-

<space> -cursor la dreapta
<caps-O> -cursor la stinga
<enter> -sfirsitul editarii cu validarea modificarilor
< Q > -sfirsitul editarii fara validarea modificarilor
< K > -sterge caracterul de sub cursor
< I > -inserare de caractere (se iese cu <enter>)
< X > -salt la sfirsitul liniei
< C > -scrie peste caracterele existente; se iese cu <enter>

* F x,y, sir1,sir2
* cauta si substituie *
-intre liniile x si y se cauta sir1 si se inlocuieste cu sir2

2 MEMORAREA, INCARCAREA SI TIPARIREA PROGRAMELOR

* P n,m;s * -salvare program



-textul sursa cuprins intre liniile n si m inclusiv va fi plasat pe caseta sub numele s

* G ,,s * -incarcare program cu numele s de pe caseta

* W,n,m,s * -salvarea unor portiuni de program
-portiunile astfel salvate pot fi inserate in programe folosind optiunea de compilare $F

- Optiuni de compilare -

Acestea pot sa apara in liniile programului la fel ca si comentariile incadrate de acolade. In momentul intilnirii lor la compilare se schimba conditiile de compilare, intrind in vigoare cele specificate de utilizator.

* {$F s } -aparitia acestor comenzi in textul programului are ca efect in momentul compilarii introducerea in acel punct a unei proceduri salvate cu W sub numele s

* {$L - } -se sisteaza tiparirea textului sursa pe ecran (rezulta cresterea vitezei de compilare)

* {$P } -comuta canalul de iesire de la TV la imprimanta sau invers

3. OPTIUNI PENTRU MODUL DE COMANDA

* CAPS-1 -revenire in modul de comanda
* B -return in BASIC (revenire cu RANDOMIZE USR 24608)
* C n -compilare incepind cu linia n
* D n,m -stergere linii
* E n -editarea liniei n
* F n,m,f,s, -cautare cu optiunile
  <s> -substituire
  <f>-fara substituire
* I n,m -inserare linii
* G ,, s -incarcare program
* K n -fixeaza numarul de linii afisate simultan pe ecran
* L n,m -listare pe ecran
* M n,m,, -copierea liniei n de pe pozitia m
* N n,m -renumerotare
* P n,m,s -salvare program
* R -lansare program (RUN)
* T -compilare si salvarea textului compilat. Se lanseaza cu RANDOMIZE USR 24608
  Atentie!! Salvarea distruge compilatorul
* V -indica forma comenzii F n,m,f,s
* W n,m,s -salvare subprograme
* X -furnizeaza adresa de sfirsit a compilatorului in hexa

4. OPTIUNE DE EDITARE

<space> -cursor dreapta
<caps-0> -cursor stinga
<caps-5> -stergerea liniei inclusiv eticheta

<caps-8> -salt la urmatorul TAB
<enter> -iesire din editare cu validarea modificarilor
<C> -overwrite
<F> -cautare
<I> -inserare
<K> -stergere
<L> -listarea liniei cu modificarile facute
<Q> -iesirea din editare fara validarea modificarilor
<R> -reincarcarea liniei fara modificari
<S> -substitutie
<X> -salt la sfirsitul liniei

5. MESAJE DE EROARE

-1- numar prea mare
-2- lipseste " ; "
-3- identificator nedeclarat
-4- lipseste un identificator
-5- s-a folosit " := " in loc de " = " in declararea constantelor
-6- lipseste " = "
-7- identificator ce nu poate aparea in membrul sting al unei atribuiri
-8- lipseste " := "
-9- lipseste ") "
-10- tip eronat
-11- lipseste " . "
-12- lipseste un factor
-13- lipseste o constanta
-14- acest identificator nu este o constanta
-15- lipseste THEN
-16- lipseste DO
-17- lipseste TO sau DOWNTO
-18- lipseste " ("
-19- incompatibilitate de tip
-20- lipseste OF
-21- lipseste " , "
-22- lipseste " : "
-23- lipseste PROGRAM
-24- lipsa variabila deoarece parametrul este de tip variabila
-25- lipseste BEGIN
-26- lipseste o variabila la utilizarea lui READ
-27- acest tip de expresii nu pot fi comparate
-28- tipul utilizat trebuie sa fie INTEGER sau REAL
-29- acest tip de variabila nu poate fi introdus (?)
-30- acest identificator nu este un tip
-31- exponentul trebuie sa fie un numar real
-32- lipseste o expresie scalara (nenumerica)
-33- nu sint permise siruri vide; folositi CHR(O)
-34- lipseste " ["
-35- lipseste "] "
-36- indicii de tablou trebuie sa fie de tip scalar
-37- lipsetse " .. "
-38- lipseste " , " sau "] " intr-o declaratie de tablou
-39- limita inferioara > limita superioara
-40- multime mai mare de 256 de elemente
-41- identificatorul de dupa FUNCTION () trebuie sa fie de tip
-42- lipseste " , " sau "] " intr-o multime
-43- lipseste " .. " sau "] " intr-o

multimi

-44- tipul parametrului trebuie sa fie un identificator de tip

-45- a nu se folosi o multime vida ca prim factor in afara unei atribuiri

-46- lipseste un tip scalar inclusiv REAL

-47- lipseste un tip scalar exclusiv REAL

-48- multimile nu sint compatibile

-49- " < " si " > " nu pot aparea in comparatii de multimi

-50- lipseste FORWARD, LABEL, CONST, VAR, TYPE sau BEGIN

-51- lipseste un numar hexa

-52- comanda POKE nu poate fi folosita pentru multimi

-53- matrice prea mare (>64)

-54- lipseste END sau " ; " intr-o declaratie de articol

-55- lipseste un identificator de cimp

-56- lipseste variabila de dupa WITH

-57- variabila de dupa WITH trebuie sa fie de tip articol

-58- identificatorul de cimp nu e precedat de WITH

-59- lipseste valoara intreaga de dupa LABEL

-60- lipseste valoarea intreaga de dupa GOTO

-61- pointer indicind un cimp eronat (?)

-62- pointer nedefinit

-63- parametrul pentru SIZE trebuie sa fie variabila

-64- se pot face doar teste de egalitate pentru pointeri

-67- singura forma de extragere a intregilor cu 2 caractere este e:m:H

-68- sirurile de caractere nu trebuie sa contina EOLN

-69- parametrii instructiunilor NEW, MARK si RELEASE trebuie sa fie variabile de tip pointer

-70- parametrii instructiei ADDR trebuie sa fie variable

6. ERORI DE EXECUTIE

1	oprire
2	depasire
4	impartire la zero
5	indice prea mare
6	indice prea mic
7	eroare intr-o rutina matematica
8	numar prea mare
9	lipseste un numar
10	linie prea lunga
11	lipseste un exponent



Program CAD (grafic) pentru simularea de aparate, instalaţii şi flux tehnologic pe calculatorul Tim-S

• C.DRUGĂRIN •
• S.RÁDULY •

Programul scris pentru calculatorul TIM-s, avind un numar dat de elemente standard (reactoare, coloane etc.), permite formarea de scheme tehnologice. Componentele standard pot fi pozitionate pe orice parte a schemei tehnologice la o scara dorita, cu posibilitatea realizarii conexiunilor dintre acestea. Eventualele erori se pot corecta, desenul obtinut se poate copia la imprimanta si se poate salva pe caseta. Programul este destinat pentru scopuri didactice, proiectari si inginerie chimica.

Prin generalizarea calculatoarelor de tip TIM-s, s-a pus la indemina invatamintului si cercetarii din tara noastra un mijloc de calcul cu calitati deosebite, care pe linga puterea de calcul apreciabila permite proiectarea si executia de desene de complexitate ridicata. Capitolul de informatica cu aplicatii de acest gen: "Grafica pe calcul tor" ("Computer Aided Design" - CAD) s-a dezvoltat intens in ultimii ani, aparitiile recente 1, 2 reflectind acest proces.

Prezenta lucrare intentioneaza sa puna la indemina chimistilor, inginerilor chimisti, proiectanti, studenti etc. preocupati de tehnologie chimica, un program care reduce efortul de proiectare/desenare a schemelor de tehnologie chimica.

In mod obisnuit, executia unui desen pe ecranul calculatorului implica scrierea unui program unic, cu cite un rind (instructiune) pentru fiecare element grafic (linie, cerc etc.) de pe figura. Astfel, o schema tehnologica nu prea complicata (coloana de rectificare), necesita un program de 50-70 de instructiuni, iar o instalatie mai complexa pina la 400-500 de instructiuni. Un program astfel scris este greu de adaptat daca se cer modifi-

cari in dimensiunile relative ale elementelor constitutive sau in pozitia absoluta sau relativa a elementelor schemei.

Prezentul program usureaza efortul de desenare-proiectare, inlaturind necesitatea cunoasterii limbajului de programare, desenul solicitat fiind usor de definit iar ulterior usor de modificat.

Elementele de desen mai frecvent utilizate (reactoare, coloane, pompe, filtre etc.) se gasesc definite de la inceput, si utilizatorul - daca se multumeste cu elementele existente - nici nu trebuie sa cunoasca modul de desenare de catre calculator a elementelor respective. (Dupa lansarea programului sint prezentate elementele predefinite in program cu denumirea si desenul corespunzator. Daca utilizatorul necesita si alte elemente, poate defini el insusi componente noi, extinzind astfel posibilitatile programului, dar pentru aceasta este necesar sa cunoasca putin elementele grafice ale limbajului BASIC).

Pentru executia unui desen, programul solicita urmatoarele date:

- numarul de elemente in desen (de ex.2 reactoare, 1 pompa, 1 filtru, in total 4 elemente);
- pentru fiecare element se va preciza:
  - -tipul (coloana, pompa, filtru etc.)
  - -pozitia (Xo,Yo coordonatele coltului stinga-jos a dreptunghiului in care se incadreaza elementul de desen. Ecranul este un dreptunghi de 255x175 de puncte)
  - -dimensiunea (Se indica -Fs- factorul de scala initial unitar cu care se inmulteste diagonala dreptunghiului de definitie (bineinteles, si fiecare segment al elementului de desen)
  - -conexiunile (conductele care intra si ies din elementul de desen respectiv). Se vor preciza:
    - -Nr.de conducte de intrare.
      - Pentru fiecare:
        - -Tipul (abur,apa,reactant)
        - -Sursa (elementul de desen de unde vine).
    - -Nr.de conducte de iesire.
      - Pentru fiecare:
        - -Tipul (abur,apa,reactant)
        - -Destinatia (elementul de desen unde se va lega)

Dupa introducerea parametrilor de desen programul de intoarce in meniu de unde se poate comanda executia desenului pe baza acestor elemente. Utilizatorul va compara figura de pe ecran cu cel prevazut si in caz de neconcordanta va modifica parametri eronati, verificind din nou executia desenului pina ce se obtine proiectul corect.

Programul permite amplasarea unor texte explicative - in masura locului liber ramas disponibil.

Desenul obtinut se poate tipari la scara 1:1 prin comanda COPY la o imprimanta grafica (ROMOM, SCAMP), se poate salva ca si fisier-imagine cu SAVE SCREEN$ si se poate salva ca si fisier de date cu SAVE DATA A().

Fisierul imagine, dupa o reincarcare ulterioara, permite obtinerea unor copii marite la scara de 2:1 sau 3:1 folosind un program utilitar grafic, iar fisierul

de date permite obtinerea intr-un timp foarte scurt a unor desene ce difera putin fata de desenul initial.

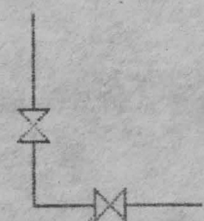
O versiune ulterioara a programului va contine partile necesare pentru comanda unui plotter si posibilitatile de decupare-vizualizare la scara marita a unor detalii de desen.

[illegible]

INTRODUCERE PARAMETRI	I
EXECUTIE DESEN	D
CORECTURA DESEN	C
SALVARE PARAMETRI	P
SALVARE ECRAN	E
OPRIRE PROGRAM	O

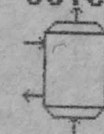
CE FACETI?

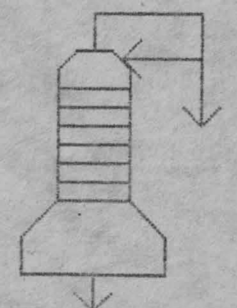
```
1 P 120 80 0
2 D 0 -30 0
3 R 120 50 1
4 D 0 -20 0
5 D 20 0 0
6 R 140 20 0
7 D 25 0 0
8 S 0 0 0
```



```
1 P 50 0 0
2 D 0 15 0
3 D -5 -5 0
4 M 5 5 0
5 D 5 -5 0
6 P 20 25 0
7 D 10 -10 0
8 D 40 0 0
9 D 10 10 0
10 D 0 70 0
11 D -10 10 0
12 D -40 0 0
13 D -10 -10 0
14 D 0 -70 0
15 D 60 0 0
16 M 0 70 0
17 D -60 0 0
18 P 50 105 0
19 D 0 15 0
20 D -5 -5 0
21 M 5 5 0
22 D 5 -5 0
23 P 0 85 0
24 D 20 0 0
25 D -5 5 0
26 M 5 -5 0
27 D -5 -5 0
28 P 20 35 0
29 D -20 0 0
30 D 5 5 0
31 M -5 -5 0
32 D 5 -5 0
33 T 15 5 0
34 S 0 0 0
```

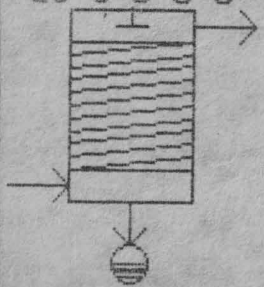
SC.CAL

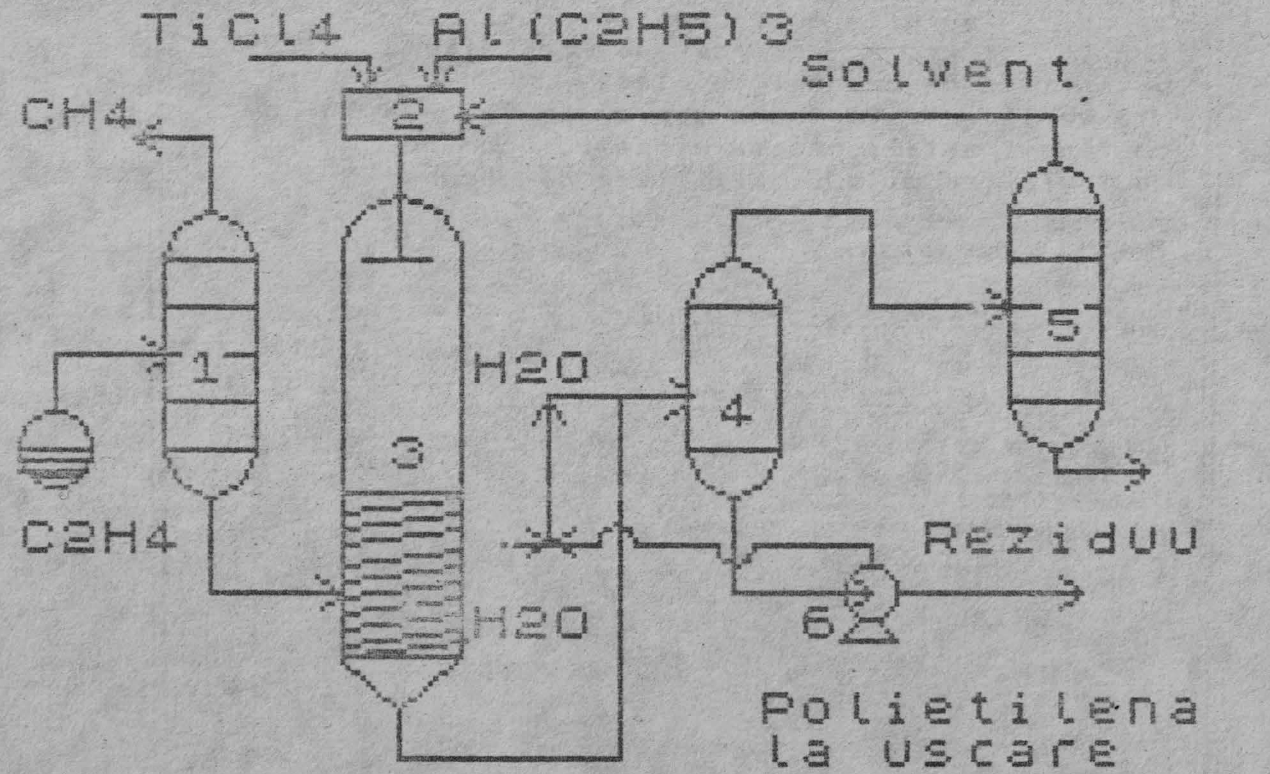




COL.RE

```
C1 P 20 25 0
 2 D 40 0 0
 3 D 0 60 0
 4 D -40 0 0
 5 D 0 -60 0
 6 H 20 35 10
 7 H 40 40 0
 8 P 0 30 0
 9 D 20 0 0
10 D -5 5 0
11 M 5 -5 0
12 D -5 -5 0
13 P 40 25 0
14 D 0 -13 0
15 D -5 5 0
16 M 5 -5 0
17 D 5 5 0
18 B 40 6 6
19 P 60 80 0
20 D 20 0 0
21 D -5 5 0
22 M 5 -5 0
23 D -5 -5 0
24 P 40 100 0
25 D 0 -20 0
26 M -5 0 0
27 D 10 0 0
28 I 1 15 0
29 S 0 0 0
```





Schema tehnologica continua pt. Polietilena de Joasa Presiune

1-Pur.C2H4 2-Pr.cat. 3-React.
4-Sep. 5-Pur.solv. 6-P.centr.

PROGRAM IN LIMBAJUL PASCAL PENTRU REZOLVAREA SISTEMELOR DE ECUATII LINIARE

• S.L. ING. VOICU MESAROS ANGHEL •
• ING. MIODRAG PUTERITY •

In urma utilizarii in practica a programului SISLIN [1], autorii au constatat ca acesta nu corespunde pe deplin sub aspectul vitezei de lucru. De exemplu, intr-o aplicatie [2] care urmarea determinarea coeficientilor curbelor Burmester in sinteza patrupozitionala, doua programe care foloseau intensiv programul SISLIN au rulat cca. 3 ore respectiv peste 6 ore.

Pornind de la acelasi algoritm, metoda lui Gauss exacta, autorii au rescris programul SISLIN in limbajul PASCAL, in implementarea acestuia pe SPECTRUM, HP4TM (versiunea 16.1).

Se prezinta in continuare rezultatele unor teste de viteza a programului SISLIN in versiunea BETA BASIC respectiv PASCAL, comparatia evidentiind si capacitatea programelor.

GRAD SISTEM	BETA BASIC	PASCAL
10	41''	2''
15	1'58''	5''
20	4'16''	10''
25	7'55''	20''
30	13'08''	33''
35	20'02''	52''
40	limitat capacit RAM	1'17''
80		10'25''

Timpii din tabel au fost obtinuti ca medii a mai multor cronometrari in cazul unui sistem aleator. Acest sistem a fost obtinut cu functia predefinita RANDOM in PASCAL si cu functia RND(255) in BETA BASIC. Ambele functii genereaza un numar pseudoaleator intre 0 si 255. Deoarece aceste functii dau rezultate mai slabe cind sint folosite in mod repetat in bucle ce nu contin operatii de intrare/iesire, s-ar putea ca in practica sa apara mici abateri fata de timpii medii prezentati.

In listingul anexat se prezinta programul SISDEM care reprezinta un exemplu de apelare a procedurii SISLIN. In afara acesteia, mai sint prezente si procedurile GETDATA si WRDATA folosite pentru a introduce de la tastatura matricea coeficientilor si vectorul termenilor liberi respectiv pentru a afisa pe ecran vectorul solutie. In aceste proceduri se utilizeaza functia predefinita CHR cu argumentul 8 care reprezinta un cod de control de tip BACKSPACE [3]. In alte implementari de PASCAL, s-ar putea ca acest cod de control sa nu mai aiba aceeasi semnificatie.

Procedura SISLIN defineste in corpul sau procedurile TRANSFOR, REZOLVA si ORDONARE a caror descriere s-a facut in [1]. Parametrii procedurii sint:

- COEF - matricea patrata a coeficientilor sistemului
- TELIB - matricea coloana a termenilor liberi
- SOL - matricea coloana a solutiilor

Primii doi parametri sint de intrare iar al treilea de iesire. Se obseva ca procedura lucreaza cu copii ale tablourilor transmise ca parametrii actuali ceea ce are doua efecte contradictorii. Pe de o parte, unul pozitiv deoarece nu se altereaza matricea coeficientilor pe parcursul rezolvarii sistemului, si unul negativ deoarece aceleasi date sint stocate de doua ori in memorie limitind spatiul alocabil variabilelor.

In cazul in care limitele memoriei devin prohibitive la rezolvarea unor sisteme mari, parametrii COEF si TELIB se pot transmite prin referinta cu cuvintul rezervat VAR [4]. Bineinteles in acest caz matricile sint alterate dar in ultima instanta ele pot fi depozitate intermediar pe banda si reincarcate dupa rezolvarea sistemului. In HP4TM acest lucru se poate realiza cu procedurile predefinite TOUT si TIN si functiile predefinite ADDR si SIZE [3].

Se poate remarca faptul ca in corpul programului SISDEM au fost definite tipurile MATPAT si MATCOL precum si constanta N. Aceasta constanta trebuie sa apara sub acest nume deoarece ea da gradul sistemului. In PASCAL nu se poate defini un tablou de dimensiune variabila, DIM a(n) ca in BASIC, si deci aceasta constanta globala este obligatorie. Cele doua tipuri au fost definite doar pentru a spori claritatea si concizia scrierii.

Pentru a putea rezolva sisteme de ecuatii foarte mari sint posibile in afara transmiterii parametrilor prin referinta inca doua artificii:

1. Pentru fericitii posesori a unei unitati de MICRODRIVE, compilarea fisierului text sursa se poate face de pe acest periferic [3]. Fisierul text nemaiocupind loc in memorie, adresa codului obiect coboara, ramanind mai mult loc pentru variabile. Aceasta alternativa e recomandata in programe mai complexe.

2. In cazuri extreme se foloseste comanda T din editor [3]. Aceasta duce la distrugerea compilatorului prin punerea codului obiect la sfirsitul modulelor de executie si la salvarea acestora pe banda. In acest mod se mareste la maximum spatiul disponibil pentru variabile, insa punerea la punct a unui program este dificila si lenta. Se recomanda la programe deja testate.

Pentru cresterea vitezei sint disponibile optiunile compilatorului O,C,S,A si I care pot fi dezactivate [3]. Acest lucru se va face doar la programe testate, altfel se poate ajunge la caderea sistemului (de operare). Retineti ca la SPECTRUM-urile standard orice cod aflat in primii 16K RAM ruleaza mai incet cu pina la 20 % din cauza accesului prioritar al chip-ului de generare a imaginii. La TIM-S aceasta pierdere de viteza se reduce pina pe la 3 %.

Ar mai fi de mentionat faptul ca precizia numerelor reale in HP4TM este mai mica decit cea din BASIC. De aceea aplicatiile in care diferentele intre coeficienti sint mai mari decit 7 ordine de marime trebuie tratate cu prudenta. Autorii vor prezenta in viitor si o metoda de perfectionare a solutiei.

Programul a fost testat si pe un calculator QL cu ajutorul compilatorului de PASCAL al firmei COMPUTER ONE versiunea 2.0. Viteza de executie a programului a ramas aceeasi in conditiile cresterii preciziei la cca. 10 cifre semnificative.

BIBLIOGRAFIE

[1] Puterity Miodrag, Rezolvarea sistemelor de ecuatii liniare, Buletinul INF, nr. 1-2, MEI, Casa Universitarilor, Timisoara, 1987

[2] Mesaros-Anghel Voicu, Puterity Miodrag, Consideratii asupra utilizarii calculului automatizat pentru determinarea curbelor Burmester folosite in sinteza pozitionala a mecanismelor cu bare, Lucrarile Simpozionului national PRASIC'86-ROBOT'86, Brasov, 11-13.12.1986

[3] ***, HP4TM, manual de utilizare

[4] Wirth N., Jensen K., PASCAL user manual and report, second edition, Springer-Verlag, New York, Heidelberg, Berlin, 1975


```
{$L-}
PROGRAM SISDEM;

{ N este o constanta globala si da gradul sitemului}

CONST N=2;

TYPE MATPAT=ARRAY[1..N,1..N] OF REAL;
     MATCOL=ARRAY[1..N] OF REAL;

VAR A:MATPAT;
    B,X:MATCOL;
    I:INTEGER;

{ Procedurile GETDATA si WRDATA sint doar pentru demonstratie}

PROCEDURE GETDATA;

 VAR I,J:INTEGER;

 BEGIN
   FOR I:=1 TO N DO
   FOR J:=1 TO N DO
    BEGIN
     WRITE('A(',I,CHR(8),',',J,CHR(8),')=');
     READ(A[I,J])
    END;
  FOR I:= 1 TO N DO
   BEGIN
    WRITE('B(',I,CHR(8),')=');
    READ(B[I])
   END
 END;

PROCEDURE WRDATA;
```

```
VAR I:INTEGER;

BEGIN
 FOR I:=1 TO N DO
 WRITELN('X(',I,CHR(8),')=',X[I])
END;

{      S I S L I N

PARAMETRII INTRARE:
 - matricea coeficientilor introdusa intii pe linie
 - matricea termenilor liberi
PARAMERU IESIRE:
 - matricea solutiilor}

PROCEDURE SISLIN(COEF:MATPAT;TELIB:MATCOL;VAR SOL:MATCOL);

 VAR Y:ARRAY[1..N] OF INTEGER;
     I:INTEGER;

 PROCEDURE TRANSFOR;

  CONST NUL=0.0;

  VAR I,J,K,L1,L2,IAUX:INTEGER;
      AUX,RMAX:REAL;

  BEGIN
   K:=N;
   REPEAT
    RMAX:=ABS(COEF[K,K]);
    L1:=K;L2:=K;
    FOR I:=1 TO K DO
     FOR J:=1 TO K DO BEGIN
      AUX:=ABS(COEF[I,J]);
      IF (RMAX<AUX) THEN BEGIN
       RMAX:=AUX;
       L1:=I;
       L2:=J
      END
     END;
    FOR I:=1 TO N DO BEGIN
     AUX:=COEF[I,L2];
     COEF[I,L2]:=COEF[I,K];
     COEF[I,K]:=AUX
    END;
    IAUX:=Y[L2];
    Y[L2]:=Y[K];
    Y[K]:=IAUX;
    FOR J:=1 TO K DO BEGIN
     AUX:=COEF[L1,J];
     COEF[L1,J]:=COEF[K,J];
     COEF[K,J]:=AUX
    END;
    IF (COEF[K,K]=NUL) THEN HALT;
    AUX:=TELIB[L1];
    TELIB[L1]:=TELIB[K];
    TELIB[K]:=AUX;
    FOR I:=1 TO K-1 DO BEGIN
     TELIB[I]:=TELIB[I]-TELIB[K]*COEF[I,K]/COEF[K,K];
     FOR J:=1 TO K DO
      COEF [I,J]:=COEF[I,J]-COEF[I,K]*COEF[K,J]/COEF[K,K]
    END;
    K:=K-1;
   UNTIL(K=1);
   IF (COEF[1,1]=NUL) THEN HALT
  END;

 PROCEDURE REZOLVA;

  VAR I,J,K:INTEGER;

  BEGIN
   SOL[1]:=TELIB[1]/COEF[1,1];
   FOR I:=2 TO N DO BEGIN
    SOL[I]:=TELIB[I];
    K:=I-1;
    FOR J:=1 TO K DO
     SOL[I]:=SOL[I]-COEF[I,J]*SOL[J];
    SOL[I]:=SOL[I]/COEF[I,I]
   END
  END;

 PROCEDURE ORDONARE;

  VAR IAUX,K,M:INTEGER;
      AUX:REAL;
      OK:BOOLEAN;

  BEGIN
   M:=N-1;
   REPEAT
    OK:=TRUE;
    FOR K:=1 TO M DO
     IF (Y[K]>Y[K+1]) THEN BEGIN
      IAUX:=Y[K];
      Y[K]:=Y[K+1];
      Y[K+1]:=IAUX;
      AUX:=SOL[K];
      SOL[K]:=SOL[K+1];
      SOL[K+1]:=AUX;
      OK:=FALSE
     END;
   UNTIL OK
  END;

 BEGIN
  FOR I:=1 TO N DO
  Y[I]:=I;
  TRANSFOR;
  REZOLVA;
  ORDONARE
 END;

{ PROGRAM DEMONSTRATIE }

BEGIN
 GETDATA;
 SISLIN(A,B,X);
 WRDATA
END.
```

PROGRAM DE REZOLVARE A ECUAȚIEI F(X)=O PRIN METODA DIFERENȚEI DE SEMN

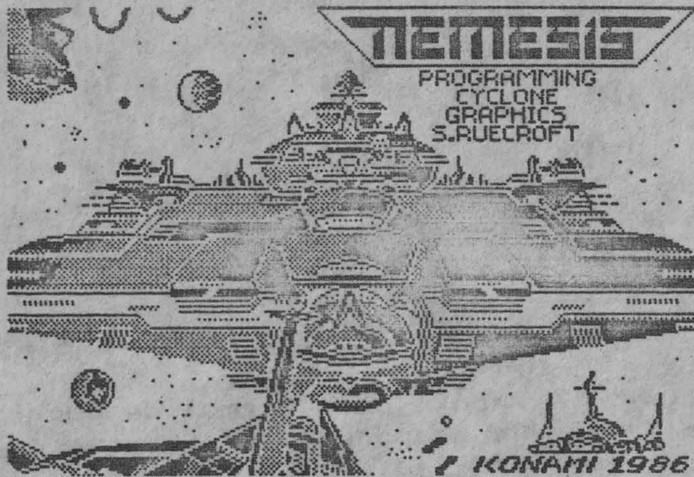
• SIMION HORAȚIU •

Programul alaturat rezolva ecuatia f(x)=O unde f(x) este o functie continua, oarecare de x. Membrul sting al ecuatiei, f(x), se introduce intre parantezele functiei SGN din linia 1000 (Pe listing f(x)=x^2 * e (-x) - 32 * PI). Programul a fost scris pentru ZX-Spectrum.

Programul cere la inceput capetele intervalului in care cauta solutiile. In program ele sint notate XI respectiv XS unde XI < x <x> S. Programul sesizeaza daca f(x) sufera un salt de semn. De aceea trebuie introdusa distante initiale dintre punctele in care se calculeaza semnul functiei, notata in program SO. Cind programul sesizeaza saltul de semn se micsoreaza acest pas de n ori si deci se intoarce la x-ul dinainte de saltul de semn unde n este numarul natural pe care utilizatorul il poate alege si pune in membrul drept al inegalitatii din linia 125. (Pe listing s-a ales pentru n valoarea 3) Deci solutia va avea exactitatea SO/IO.

Binenteles programul nu sesizeaza radacinile de ordin par, deci in care functia are un punct de extrem d, programul sesizind doar radacinile de ordin impar. Dar radacinile de ordin par ale lui f(x) sint radacini de ordin impar ale ecuatiei df/dx = O. Deci dupa ce rulam programul pentru f(x) vom rezolva cu el si ecuatia df/dx = O. Dintre solutiile din urma putem gasi, calculind valoarea lui f(x) in aceste puncte, si radacinile de ordin par ale ecuatiei noastre. In continuare se prezinta programul.

```
  5 PRINT "Program de rezolvare
 a ecuatiei f(x)=0 prin metoda d
iferentei de semn.Functia f se i
ntroduce intre parantezele SGN-u
lui de la linia 1000": FOR f=1 T
```




```
0 100: NEXT f: CLS
  10 DIM s(1000)
  14 PRINT "introduceti limitele
 argumentului si pasul initial d
e parcurgere a intervalului"
  15 INPUT xi,xs
  20 LET x=xi: LET i=1
  25 GO SUB 1000
  30 LET n=1
  33 INPUT so
  34 CLS
  35 LET e=0: LET s=so
  45 LET x=x+s: LET i=i+1
  55 IF x>xs THEN GO TO 1010
  60 PRINT AT 1,1;x;"
"
  70 GO SUB 1000
  77 IF s(i)=0 THEN GO TO 87
  81 IF s(i)*s(i-1)>0 THEN GO TO
 45
  85 IF s(i)*s(i-1)<0 THEN GO TO
 110
  86 GO TO 35
  87 LET n=n+1
  89 PRINT AT n,1;"x";n-1;"=";x
  90 GO TO 45
 110 LET i=i-1: LET x=x-s: LET e
=e+1
 125 IF e>3 THEN GO TO 140
 130 LET s=s/10: GO TO 45
 140 LET n=n+1: PRINT AT n,1;"ap
rx";n-1;"=";x
 147 LET i=i+1: LET x=x+s: GO TO
 35
1000 LET s(i)=SGN (EXP (-x)*x^2-
32*PI)
1009 RETURN
1010 PRINT AT n+ ,1;"EOJ"
```



PROGRAME DE REPREZENTARE IN PERSPECTIVA A SUPRAFETELOR

Simon Horațiu

Programele prezentate in acest articol se ocupa cu reprezentarea in perspectiva a suprafetelor, din spatiu, definite explicit. Programele difera intre ele prin tipul de coordonate in care sint definite suprafetele. Astfel:

- in reper cartezian, reprezinta suprafete definite functii de tipul:
  $z = z(x,y)$ cu $a <= x <= b$, $c <= y <= h$
- in reper sferic, reprezinta suprafete definite functii $r = r(\theta,\varphi)$
- in reper cilindric, reprezinta suprafete definite functii $z=z(r,\theta)$ s.a.m.d.

unde (x,y,z), (r,θ,φ), (z,r,θ) sint coordonatele punctelor de pe suprafete iar z, respectiv r, sint functii definite pentru orice valoare a celorlalte doua coordonate.

Programul permite realizarea, pe ecran, a imaginilor acestor suprafete, privite din orice punct al spatiului.

Realizarea perspectivei

Presupunem ca avem o suprafata definita, in modul cel mai simplu, de functia $z = z(x,y)$, in raport cu sistemul de coordonate Oxyz. De asemenea, presupunem ca ochiul pe care-l asimilam cu o lentila convergenta de distanta focala f, se afla la coordonatele sferice d,θ,φ in raport cu aceleasi axe Oxyz (vezi fig.1). Trebuie sa gasim legatura dintre cele 3 coordonate ale unui punct de pe suprafata si cele doua coordonate ale imaginii sale, care presupunem ca se formeaza in planul focal. (Deci am presupus automat ca d >> decit distanta maxima de la origine la orice punct al suprafetei).

Vom considera acum, pe linga coordonatele unui punct P in raport cu sistemul de axe Oxyz anume: $z=z(x,y),x,y$ si pe acelea in raport cu un sistem de axe Ox'y'z'. Acestea se obtin prin rotirea sistemului de axe Oxyz astfel incit axa Oy' sa coincida cu axa origine-ochi astfel incit axa Oz' sa fie continuta in planul determinat de axele Oz si Oy'. In acest caz, automat axa Ox' se afla in planul Oxy. Sistemul Ox'y'z' se obtine prin efectuarea unei rotatii cu unghiul

φ-PI/2 in jurul axei Oz si apoi prin rotire cu unghiul PI/2-θ in jurul axei Ox' (fig.1). Daca (x'y'z') sint coordonatele lui P in noul sistem si matricea de rotatie este:

$$A=\begin{pmatrix}1 & 0 & 0\\ 0 & \sin\theta & \cos\theta\\ 0 & -\cos\theta & \sin\theta\end{pmatrix}\cdot\begin{pmatrix}\sin & -\cos & 0\\ \cos & \sin & 0\\ 0 & 0 & 1\end{pmatrix}$$

obtinem legatura

$$A\begin{pmatrix}x\\ y\\ z\end{pmatrix}=\begin{pmatrix}x'\\ y'\\ z'\end{pmatrix}$$

Deci:

x' = x*sinφ - y*cosφ

y' = x*sinθ*cosφ + y*sinθ*sinφ + z*cosθ

z' = -x*cosθ*cosφ - y*cosθ*sinφ + z*sinθ

Daca in planul focal (vezi fig.2) alegem doua axe de coordonate: Oye ,axa verticala, adica paralela cu Oz', si axa Oxe perpendiculara pe aceasta, atunci coordonatele imaginii se obtin prin aplicarea legii a doua a lentilelor, de fapt niste raporturi in triunghiuri asemenea:

$$xe= - \frac{-x'*f}{d-y'}$$

unde f este distanta focala a lentilei (distanta de la ochi la planul focal); d-y' este distanta de la planul paralel cu planul z'Ox' care contine punctul P, la ochi.

$$ye= \frac{z'*f}{d-y'}$$

Deci aceasta este legatura dintre coordonatele punctului P: x,y si z=z(x,y) si coordonatele imaginii sale xe si ye, din planul focal al lentilei (cu care am aproximat ochiul). Legatura aceasta se obtine cu ajutorul coordonatelor x',y',z' ale punctului P in sistemul de axe rotit Ox'y'z'. Din ultimele doua relatii se observa ca toate punctele care se afla pe o aceeasi dreapta, din fasciculul de drepte ce trece prin ochi, dau aceeasi imagine, ceea ce concorda cu propagarea rectilinie a luminii.

Programul

Programul realizeaza trasarea imaginilor unor curbe de pe suprafata pe planul focal al lentilei care acum coincide cu ecranul.

Daca suprafata e usor de definit in reper cartezian, atunci la linia 1000 vom avea explicitat pe z:

```
1000 LET Z=.....(f(x,y))  ==> la linia
              1000 se defineste functia z!
```

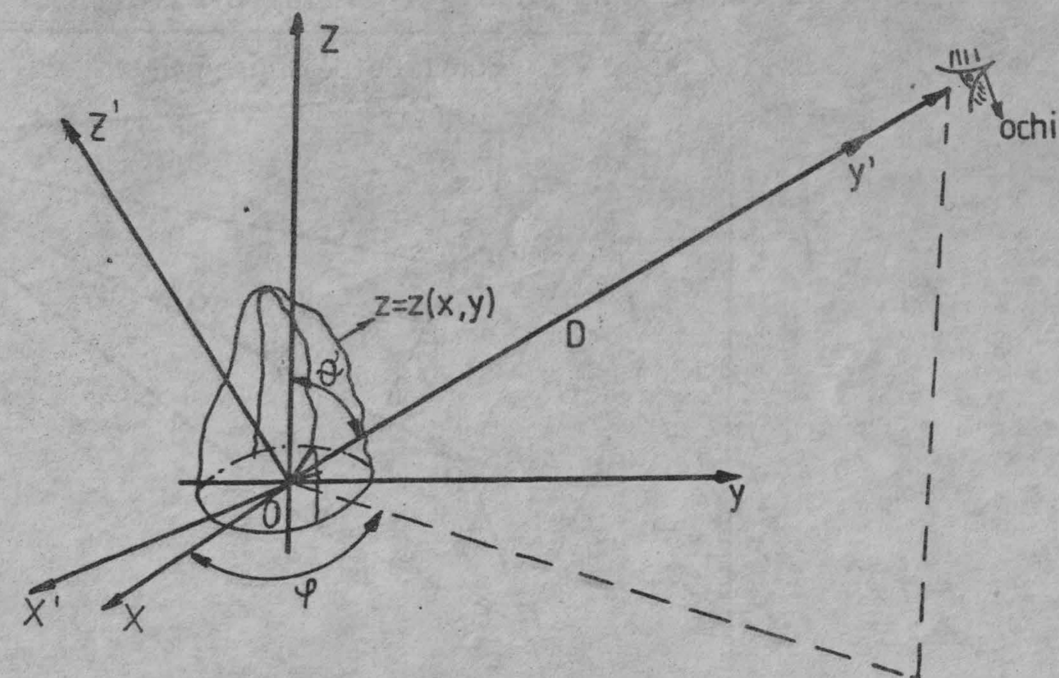


Fig. 1

In ac st caz programul va trasa pe ecran imaginea unor linii de caroiaj. Pentru aceasta, la inceput el ne cere numarul de curbe de caroiaj de pe suprafata (notat nc in program) si numarul de puncte intre care traseaza fiecare curba de caroiaj (notat np in program). In continuare se cere, prin linia 23, coordonatele ochiului in raport cu sistemul de axe de definire a suprafetei (notate d pentru d, t pentru θ, f pentru φ).

Prin linia 35, programul cere limitele intre care variaza coordonatele x si y, limite notate a,b,c,h, unde a<=x<=b si c<=y<=h, dupa care se trece la operatia de trasare. In ciclul 100-160 genereaza o retea de nc x np puncte in felul urmator: ciclul exterior alege nc valori echidistante ale lui x, intre a si b, luind inclusiv pe acestea ca valori pentru x. In ciclul interior se iau, pentru un x fixat, np valori ale lui y intre c si h. Pentru fiecare x si y, subrutina 1000-1250, care este "inima" programului, calculeaza pe z, apoi calculeaza coordo-

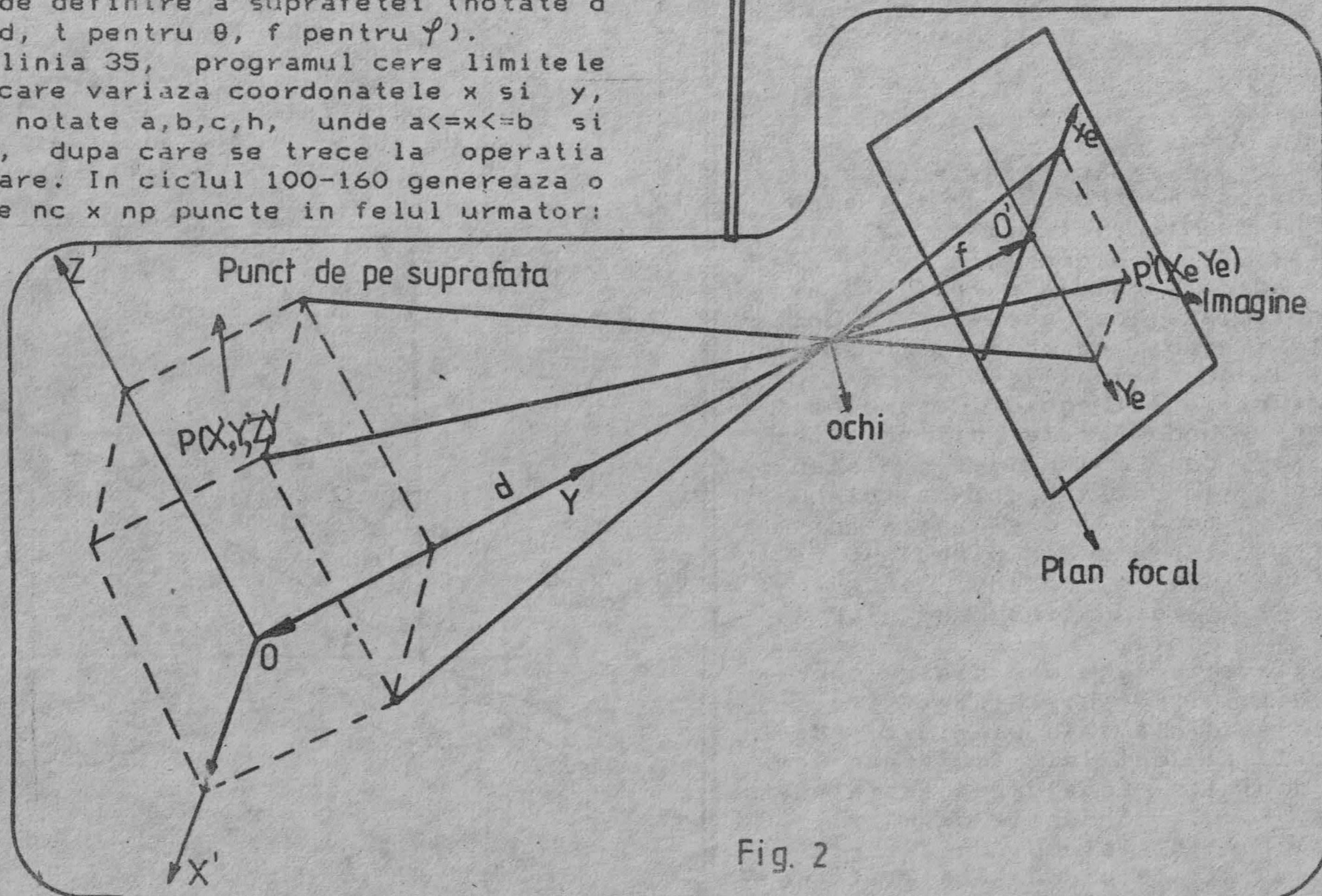


Fig. 2

natele x´,y´,z´ ale punctului de pe suprafata si in fine pe xe si ye, coordonatele imaginii (luind pentru distanta focala valoarea 1). Coordonatele imaginilor acestor puncte, care se afla pe curbe paralele cu planul y0z, se inmagazineaza in tabloul q, respectiv w. De asemenea ciclul interior face trimitere la subrutina 1250-1450, o subrutina de minim-maxim, care va calcula intre ce valori variaza coordonatele punctelor imagine din "planul focal", furnizindu-ne dupa rularea ciclurilor 100-160 si 170-235 pe: mix, max, miy, may unde mix<=xe<=max, miy<=ye<=may, oricare ar fi punctul imagine.

Similar, ciclul 170-235 va genera o retea de nc x np puncte corspunzatoare la nc curbe paralele cu planul x0z si va inmagazina cele nc x np seturi de coordonate xe si ye in tablourile e si r.

Liniile 240-270, tinind cont de cele 4 variabile furnizate de SUB 1250, asigura incadrarea imaginii intr-un patrat de 176x176 UG (unitati grafice) de pe ecran, prin calcularea unui factor de scala notat df. Cu acest df, ciclurile 276-290 si 295-315 vor trasa efectiv curbele pe ecran, folosindu-se de tablourile q, w, e si r in care s-au inmagazinat coordonatele punctelor imagine.

Dreptunghiul de 80x176 UG ramas disponibil in dreapta, e folosit, prin liniile 2000-2200, pentru "conversatie", calculatorul putind redesena imaginea cu un factor de scala mai mic decit df-ul furnizat de liniile 240-270.

Pentru a ne "roti" sau "indeparta" de suprafata, va trebui ca la o aceeasi linie 1000 si la aceleasi limite pentru x si y, sa rulam din nou programul cu alte coordonate ale ochiului.

Alte variante

I Pentru a trasa curbe definite in reper sferic de functii r=r(θ, φ), se vor aduce programului de pe listing modificarile cuprinse in lista de linii din continuarea acestuia.

II Varianta de program de pe listing se poate folosi si pentru suprafete definite in reper cilindric de functia z=z(r,θ) astfel: daca z=f´(r,θ)

1.La linia 1000 se introduce functia f´ explicitata cu variabila r notata cu x si variabila θ notata cu y si se adauga:

1000 LET z=...(f´(x,y)): LET aa=x: LET BB=y: LET xx=AA*COS(BB): LET YY=AA*SIN (BB)

2.In liniile 1010-1020 se inlocuieste x cu xx si y cu yy in membrii drepti ai LET-urilor.

3.Se introduc, prin linia 35, urmatoarele date: 0 si valoarea maxima a lui r, ca limite ale lui "x" si 0 si 2*PI ca limite ale lui "y".

III De asemenea, se pot reprezenta functii definite nu peste tot. De exemplu, functia z=arcos(α-cos(x)-cos(y)) cu |α|<3 nu este definita pentru orice x si y. Pentru aceasta se defineste tabloul m(2,nc,np) care pentru fiecare din cele 2*nc*np puncte ia valoarea 0 daca functia nu este definita si 1 daca functia este definita , pentru

valorile curente ale lui x si y. Acest tablou va interveni apoi si in ciclurile 276-315, in trasarea curbelor.

IV Tot cu un tablou suplimentar se pot elimina punctele ce "nu se vad" din suprafata. Elemetele din acest tablou iau valoarea egala cu semnul produsului scalar dintre vectorul ochi-punct si vectorul normal la suprafata in punctul respectiv. Daca acest produs scalar este negativ, atunci punctul "se vede" si cu ajutorul unei instructii IF se va trasa curba prin acest punct. Daca produsul este pozitiv, atunci punctul "nu se vede".

Ecranul 1 se obtine pentru programul de pe listing, deci pentru z=2*(x*x-y*y). Numarul de curbe a fost ales nc=15, iar numarul de puncte pe curba np=15. S-au introdus "coordonatele ochiului" d=10, θ=0.8, φ=0.4.
Dupa 6 minute a aparut pe ecran figura prezentata.

Ecranul 2 se obtine in varianta de program pentru suprafete definite in coordonate cilindrice conform celor de la punctul II. Suprafata are simetrie de rotatie in jurul axei Oz si e definita prin: z=4*(r^5/5 - 3*r^4/2 + 11*r^3/3 - 3*r^2).
S-au introdus ca limite ale lui x limitele lui r 0<=r<=3, iar ca limte ale lui y, limitele lui θ 0<=θ<=2*PI. Coordonatele ochiului au fost d~10, θ ochi ≈ 0.3-0.4 si φ=1. S-a introdus nc=15 si np=20.

Ecranele 3 si 4 reprezinta orbitale ale atomului de H si necesita preparative mai laborioase pentru realizarea lor.

NOTA.Programul poate da la un moment dat mesaj de eroare cind trece la reprezentarea grafica efectiva, la liniile 276...Pentru un numar mare de puncte np se intimpla ca unul din argumentele instructiunii DRAW sa fie prea mare din cauza rotunjirilor succesive pe care le comporta aceasta instructiune. Acest lucru se poate remedia micsorind factorul de scala df direct, dupa cum urmeaza: LET df=0.98*df: GO TO 273. Daca din nou apare mesaj de eroare, se repeta instructia de mai sus.

In incheiere as dori sa multumesc pentru sprijinul acordat tovarasilor asist. T.Siclovan de la Facultatea de Tehnologie Chimica si s.l.dr. Nagy Iosif de la Institutul de Medicina.

Modificari ce trebuie aduse programului de mai sus pentru a reprezenta suprafete definite in coordonate sferice de functii r=r(θ,φ).

```
1000 LET s(i) =SGN ((x-1)*(x-2)*(x-3))
1009 RETURN
1015 PRINT AT n+3,1; "Intervalul a fost
parcurs"
```

```
  1 PRINT AT 8,6;"Program de re
prezentare in perspectiva a supr
afetelor z=z(x,y).La linia 1000
se introduce functia z.": FOR i=
1 TO 10: PAUSE 40: NEXT i: CLS
  2 REM td
  3 PRINT "Introduceti numarul
de curbe de caroiaj a suprafetei
 si apoi numarul de puncte intre
 care se traseaza o curba de car
oiaj."
  4 INPUT nc,np
  5 DIM q(nc,np): DIM w(nc,np):
 DIM e(np,nc): DIM r(np,nc)
```

```
 22 CLS
 23 PRINT "Introduceti coordona
tele sferice ale"" ochiului"" in
 raport cu sistemul de axe de de
finitie a suprafetei: distanta o
chi-origine,latitudinea,longitud
inea"
 25 INPUT d,t,f
 27 CLS

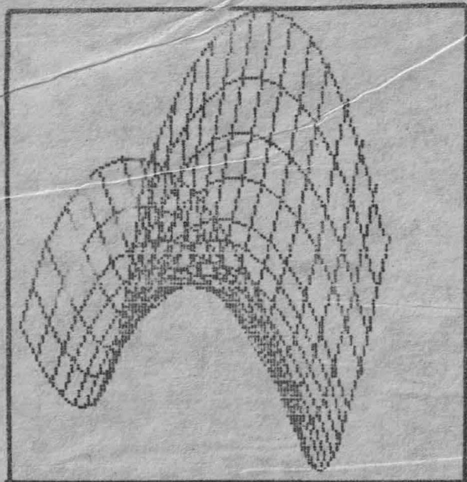
 30 PRINT "Introduceti limitele
 inferioara apoi superioara pent
ru coordonata x .Repetati pentru
 coordonata y."
 35 INPUT a,b,c,h
 40 CLS
 50 LET x=a: LET y=c
 65 GO SUB 1000
 75 LET mix=xe: LET max=xe
 80 LET miy=ye: LET may=ye
100 FOR j=1 TO nc
103 LET x=a+(j-1)*(b-a)/(nc-1)
105 FOR k=1 TO np
108 LET y=c+(k-1)*(h-c)/(np-1)
110 GO SUB 1000
115 GO SUB 1250
149 LET q(j,k)=xe
150 LET w(j,k)=ye
155 NEXT k
157 BEEP 0.3,1
160 NEXT j
170 FOR k=1 TO nc
173 LET y=c+(k-1)*(h-c)/(nc-1)
175 FOR j=1 TO np
```

```
177 LET x=a+(j-1)*(b-a)/(np-1)
180 GO SUB 1000
185 GO SUB 1250

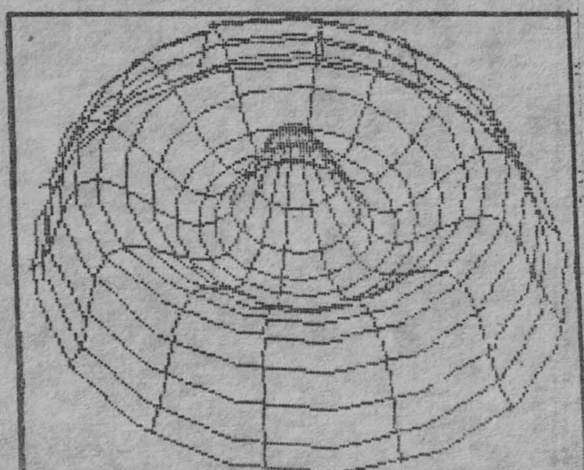
 222 LET e(j,k)=xe: LET r(j,k)=y
e
 225 NEXT j
 230 BEEP .3,2
 235 NEXT k
 240 LET dfx=170/(max-mix)
 250 LET dfy=170/(may-miy)
 255 IF dfx<=dfy THEN GO TO 270
 260 LET df=dfy
 265 GO TO 273
 270 LET df=dfx
 273 CLS
 276 FOR j=1 TO nc
 277 PLOT 4+(q(j,1)-mix)*df,4+(w
(j,1)-miy)*df
 280 FOR k=2 TO np
 282 DRAW (q(j,k)-q(j.k-1))*df,(
w(j,k)-w(j,k-1))*df
 285 NEXT k
 290 NEXT j
 295 FOR k=1 TO nc
 297 PLOT 4+(e(1,k)-mix)*df,4+(r
(1,k)-miy)*df
 300 FOR j=2 TO np
 305 DRAW (e(j,k)-e(j-1,k))*df,(
r(j,k)-r(j-1,k))*df
 310 NEXT j
```

```
 315 NEXT k
 365 GO TO 2000
1000 LET z=2*x*x-2*y*y
1010 LET xp=SIN (f)*x-COS (f)*y
1015 LET yp=SIN (t)*COS (f)*x+SI
N (t)*SIN (f)*y+COS (t)*z
1020 LET zp=-COS (t)*COS (f)*x-C
OS (t)*SIN (f)*y+SIN (t)*z
1050 LET xe=-xp/(d-yp)
1055 LET ye=zp/(d-yp)
1060 RETURN
1250 IF xe>=mix THEN GO TO 1300
1270 LET mix=xe
1300 IF xe<=max THEN GO TO 1350
1330 LET max=xe
1350 IF ye>=miy THEN GO TO 1400
1370 LET miy=ye
1400 IF ye<=may THEN GO TO 1450
1420 LET may=ye
1450 RETURN
2000 PLOT 0,0
2005 DRAW 0,175
2010 DRAW 255,0
2015 DRAW 0,-175
2020 DRAW -255,0
2025 PLOT 175,0
2030 DRAW 0,175

2040 PRINT AT 2,23;"continue?":
PRINT AT 3,23;"da1nu0"
2045 INPUT k
2050 IF k<>0 AND k<>1 THEN GO TO
 2045
2055 IF k=0 THEN GO TO 2200
```

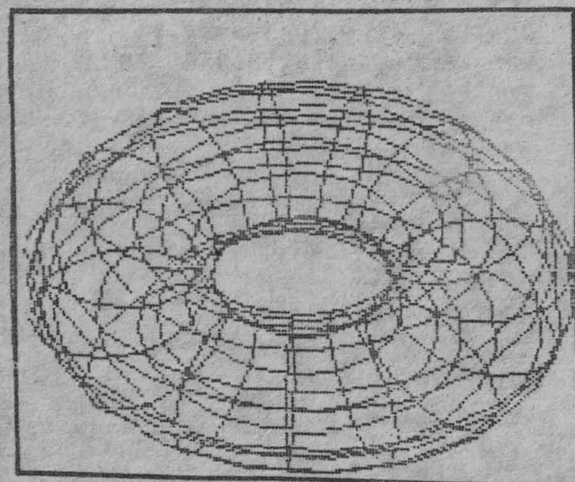


```
2060 PRINT AT 5,23;"micsorez": P
RINT AT 7,23;"doar?"
2065 INPUT k
2070 IF k<>0 AND k<>1 THEN GO TO
 2065
2075 IF k=0 THEN GO TO 2
2080 PRINT AT 9,23;"de cite": PR
```



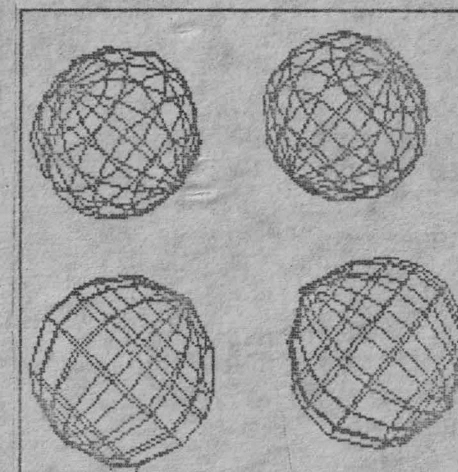
```
INT AT 11,23;" ori?"
2085 INPUT ms
2090 IF ms<1 THEN GO TO 2085
2095 LET df=df/ms
2097 GO TO 273
2200 PRINT AT 13,23;"EOJ"
```

```
   1 PRINT AT 8,6;"Program de re
prezentare in perspectiva a supr
afetelor definite in coordonate
sferice r=r(teta,fi).Introduceti
 functia r=r(t,f) LA LINIA 1000.
```



```
": FOR f=1 TO 10: PAUSE 40: NEXT
 f: CLS
  25 INPUT d,tp,fp: REM acestea
vor fi noile notatii ptr.coordon
atele ochiului
  30 REM liniile 30,35,40 dispar
  50 LET t=0: LET f=0
```

```
 103 LET t=0.1+(j- )*2.942/(nc-1
)
 108 LET f=(k-1)*2*PI/(np-1)
 173 LET f=(k-1)*2*PI/(nc-1)
 178 LET t=(j-1)*PI/(np-1).
1000 LET rp=20
1010 LET x=rp*SIN (t)*COS (f)
1015 LET y=rp*SIN (t)*SIN (f)
1020 LET z=rp*COS (t)
1030 LET xp=SIN (fp)*x-COS (fp)*
y
1035 LET yp=SIN (tp)*COS (fp)*x+
SIN (tp)*SIN (fp)*y+COS (tp)*z
1040 LET zp=-COS (tp)*COS (fp)*x
-COS (tp)*SIN (fp)*y+SIN (tp)*z
```



ATARI ST

• DRAGOMIR RADU •

Este de fapt o familie de calculatoare dintre care cel mai vechi reprezentant este modelul 260 ST. Apoi a apărut 520 ST. In jurul lui s-a făcut foarte multă vîlvă. In anul 1985 el a fost declarat **Calculatorul anului.** Comisia care atribuie asemenea titlu este compusă din mai **mulți** membri din U.S.A. , U.K. , R.F.G. , Italia, Japonia, Spania, Iugoslavia, Franța, Ungaria, Polonia. Fiecare din aceștia, pe baze statistice alcătuiesc cîte un *clasament al vînzărilor* dintr-un an pentru calculatoarele ce se gasesc în magazine. In final se stabilește un clasament general. Primul clasat este declarat **calculatorul anului.** De fapt, dacă analizăm mai bine, ce înseamnă **calculatorul anului**, ne putem da seama că el reprezintă cel mai bine vîndut calculator al anului, adică acela care are raportul preț/performanță cel mai scăzut.

Performanțele unui calculator se pot măsura doar prin intermediul unor factori ca:

- structură HARD
- ușurință în folosire
- posibilități de dezvoltare (HARD, SOFT)
- compatibilitate (HARD, SOFT)
- fiabilitate

Să analizăm pe rînd pe fiecare din acești factori

Structura HARD

Construit în jurul unui microprocesor dintre cele mai bune din lume **MOTOROLA 68000.** Este un microprocesor ce lucrează intern pe 32 de biți, iar extern pe 16 biți. El poate adresa 16 Mbytes. La calculatoarele ATARI ST memoria RAM variază de la un model la altul între 0.25 și 16 Mbytes. Are o memorie ROM de 192 Kbytes. Are 3 moduri de afișare a informației:

rezoluție 640x400 puncte în regim monocrom
rezoluție 640x200 puncte în regim 4 culori
rezoluție 320x200 puncte în regim 16 culori

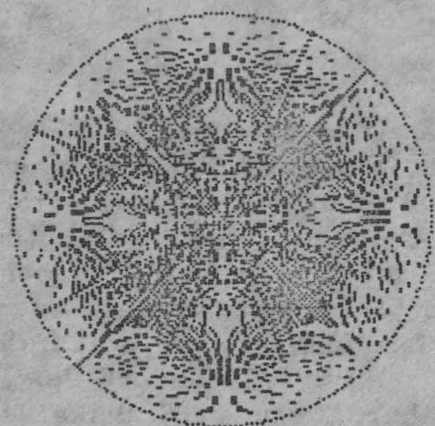
Culorile sînt la nivel de punct și se pot selecta la alegere dintr-o paletă de 512 nuanțe. In ceea ce privește sunetul există 3 canale independente ce pot furniza semnale în intervalul 30 - 16000 Hz. Ca suport de memorare se pot folosi floppy diskurile sau hard diskul (se pot conecta 2 unități de floppy disk și o unitate de hard disk. ATARI promovează lucrul cu unități floppy de 3.5" , dar interfața de floppy disk (Floppy Drive Controller) este standardizată astfel încît se pot cupla orice fel de unități de floppy, deci și de 5.25".

Ușurință în folosire

Inițial sistemul de operare a fost TOS (echivalent cu CP/M68), dar i s-a adăugat o interfață cu utilizatorul denumită **Graphics Environment Manager** , prescurtat **GEM**. Această interfață a fost concepută astfel încît să ușureze cît mai mult munca utilizatorului, oferindu-i facilitățile obișnuite ale unui sistem de operare (gen SFDX, CP/M, MS-DOS, etc.) precum și o siguranță sporită. Ușurința constă în faptul că utilizatorul începator nu se va speria de lucrul cu acest calculator. Nu va fi nevoit să țină minte comenzi destul de abstracte de genul

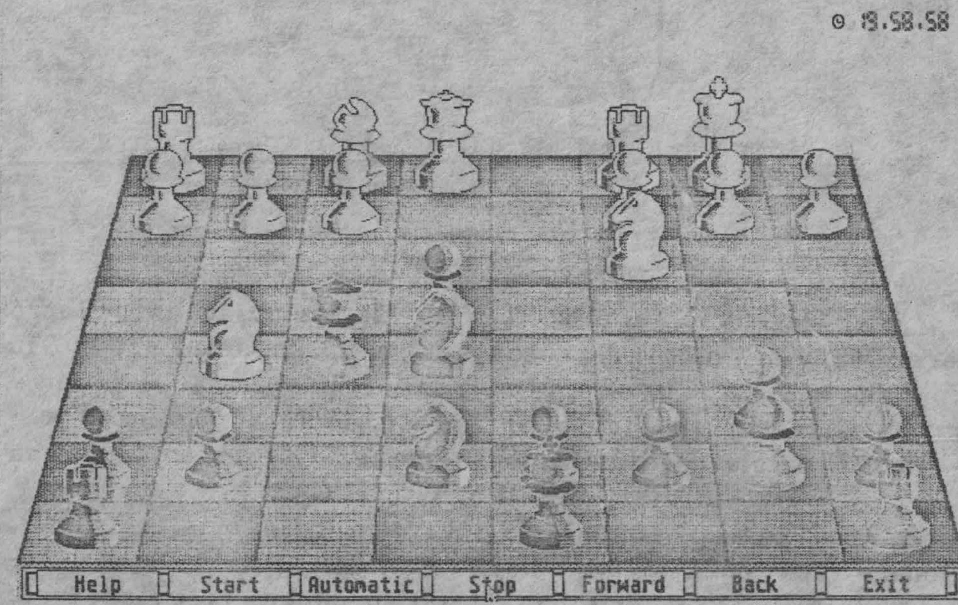
```
DIR
FORMAT A:/S/V
DIP B:=A:PROGRAM.BAS[PQV]
```

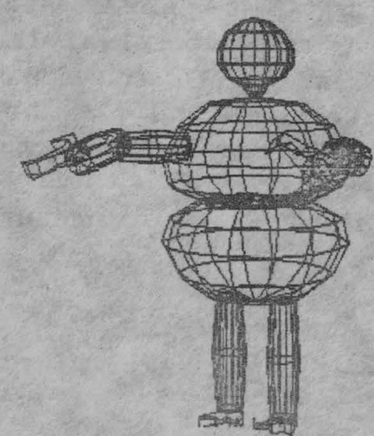
care lui nu îi spun pentru început nimic. Ecranul noului sistem arată cam așa

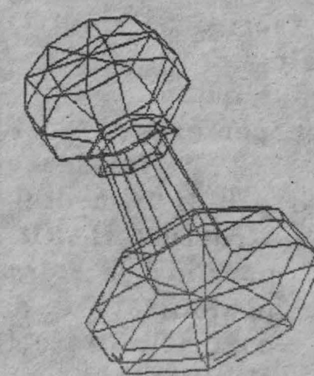


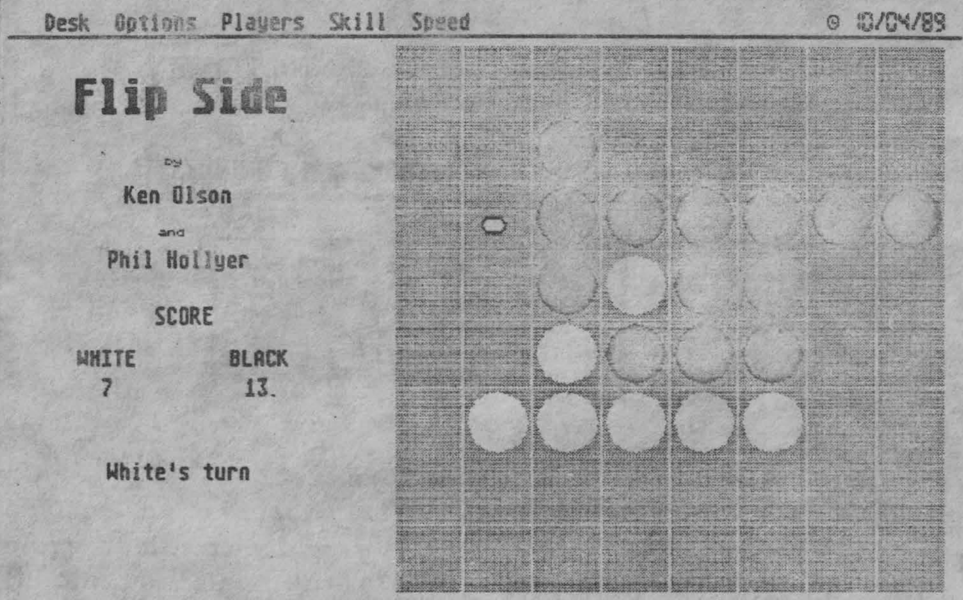
ASSEMBLER PROFIMAT

Există de asemenea programe de grafică PAINTER, Grafic3D, editoare de texte: Tempus WordWriter, WordPlus, Signum2, gen bază de date: Data Manager, Swift Calc, jocuri nu foarte multe: Bounce, Snake, Flight Simulator 2, Chess, Flip Side, Arkanoid, Alternate Reality, The Sentinnel, programe utilitare de tot felul.







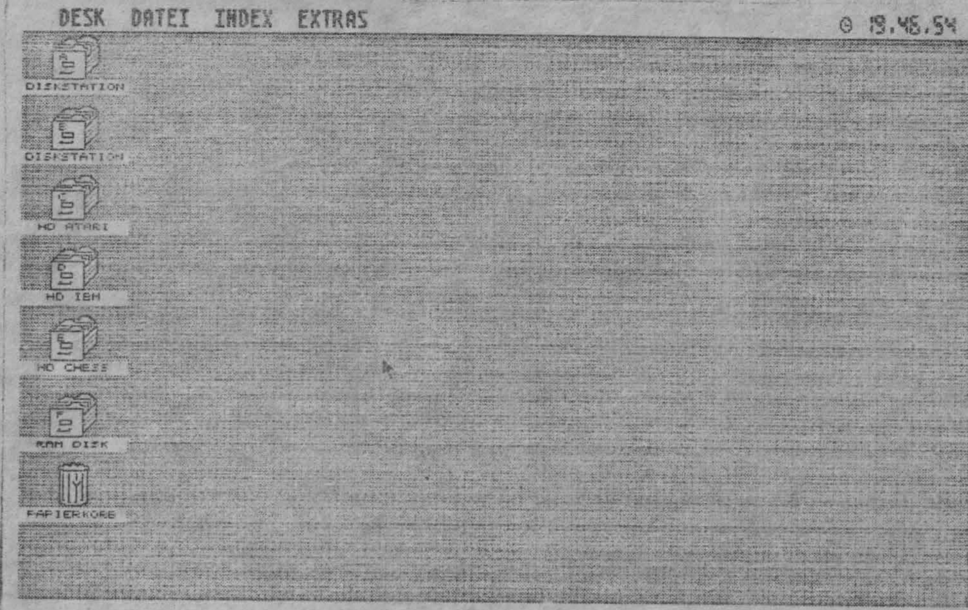


FIABILITATE

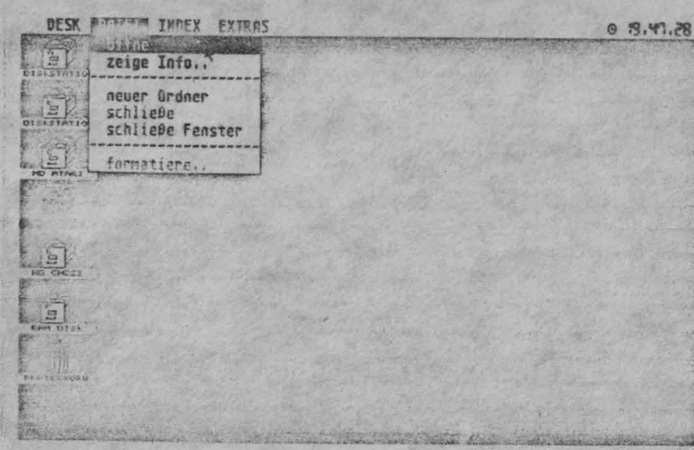
In ceea ce privește fiabilitatea nu pot face decît o singură afirmație: *Calculatorul ATARI ST este un calculator care ține la tăvăleală !*

Iată cîteva ditre argumentele datorită cărora 520 ST a fost declarat calculatorul anului în 1985, iar 1040 ST în 1987. După modelul 520 ST au mai apărut în ordine 1040 ST, 2080 ST, 4160 ST. Diferențele dintre acestea constau doar în capacitatea de memorie RAM care este de 1Mbytes, 2Mbytes, respectiv 4Mbytes, față de 520 ST care are doar 0.5Mbytes. Modificări minime au fost aduse (de fapt îmbunătățiri și eliminarea unor neajunsuri) și au apărut modelele MEGA ST2 și MEGA ST4. Recent a apărut (sfîrsitul lui 1988) un nou ST compatibil, dar mult mai pretențios (tehnologie AT) ATARI TT (TT/X).

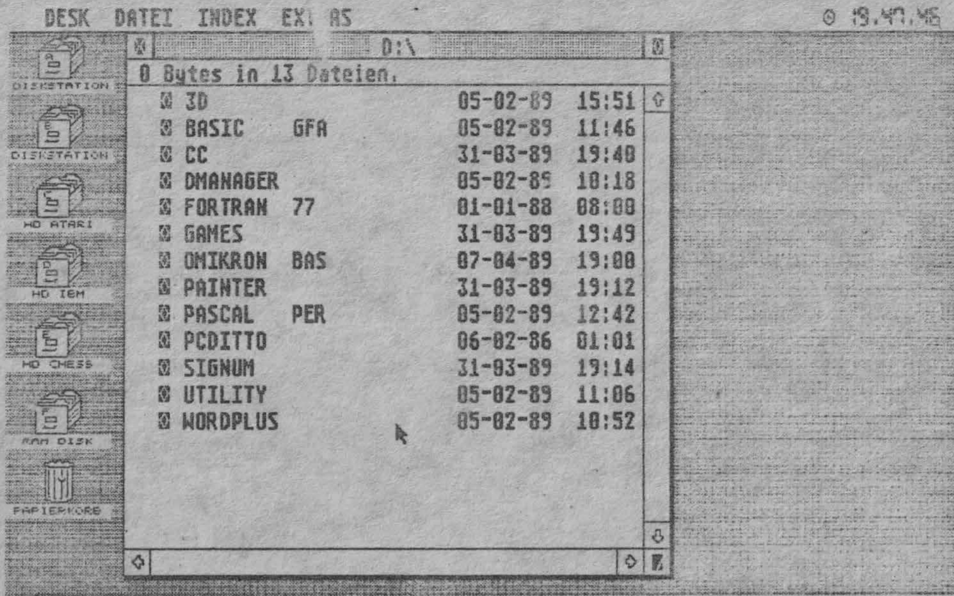
Notă: editarea acestui articol a fost făcută pe un ATARI 1040 ST cu editorul SIGNUM2, folosind o imprimantă NEC P7, din dotarea Clubului Sportiv Scolar nr 1 Timișoara.



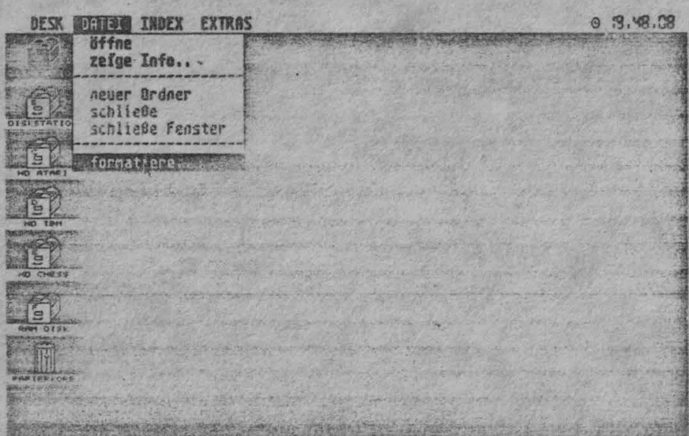
Sertarele reprezintă unitățile logice existente în sistem. Acestea sînt sertarele unui fisier (vechea imagine a unui fișier). Ele conțin multe fișe cu informații utile. Cine nu știe că dacă dorește să vadă ce este într-un sertar trebuie să-l deschidă (lucru pe care, după ce am indicat sertarul, îl putem face alegînd opțiunea **öffne** din meniu.



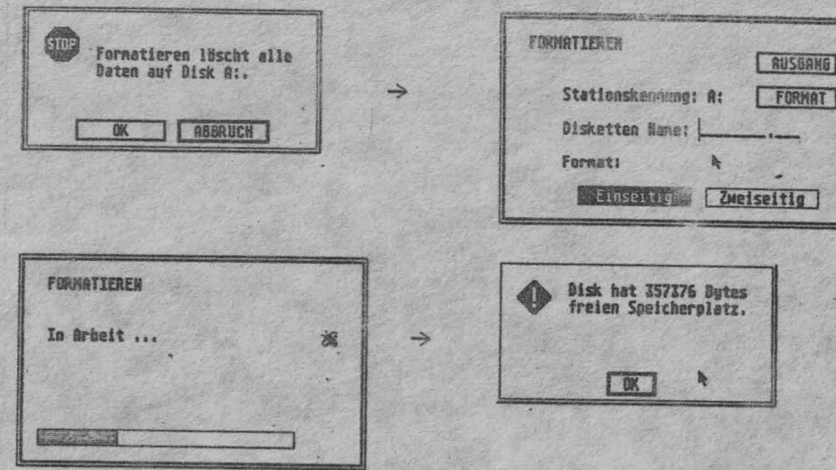
și am obținut :



adică exact ceea ce conține sertarul cu pricina. Așa se poate da o comandă gen DIR pe calculatorul ATARI ST. Formatarea unui disk decurge în felul următor: se selectează sertarul și se alege opțiunea **formatiere** . .



... după care urmează un dialog calculator-utilizator care are drept scop prelurea parametrilor



... și în sfîrșit se obține un disk formatat.

Pe ecranul noului sistem mai puteți observa un coș de gunoi. La coș se aruncă fișierele de care nu mai avem nevoie (așa se șterg fișierele). Celor obișnuiți cu modul clasic de lucru s-ar putea să li se pară un mod greoi de lucru, dar nu e așa. Mulți, din ce în ce mai mulți îl preferă pe acesta, chiar și cei care îl realizează. Toate aceste facilități de lucru cu meniuri, ferestre, icon-uri, dialog box-uri sînt scrise în ROM (din această cauză este așa de mare 192Kbytes). Limbajele de programare existente pentru ATARI ST sînt dotate toate cu funcții și proceduri care fac apel la cele din ROM. Deci punerea la punct a unor programe bune, ușor de manevrat este într-adevăr floare la ureche! Toate programele scrise pentru ATARI ST au urmat aceeași idee, astfel încît toate sînt la fel de ușor de utilizat.

<u>Compatibilitatea</u>

Deoarece acest nou mod de lucru este foarte diferit de cele anterioare ne putem da ușor seama că : **ADIO COMPATIBILITATE** ST-urile sînt compatibile între ele și atît. Da, acest lucru este adevărat, sau mai bine zis . . .

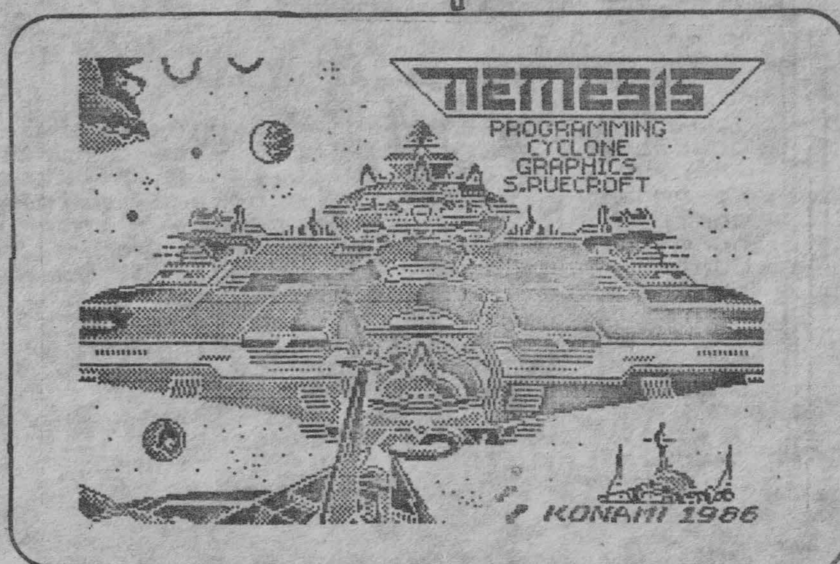
a fost adevărat pînă cînd niște programatori mai îndrăzneți și bine plătiți au facut primul emulator pentru ATARI ST. *Ce este un emulator ?* Este un program ca oricare altul scris pentru calculatorul pe care acesta trebuie să ruleze. *La ce servește ?* Oferă utilizatorului compatibilitatea cu alte <u>sisteme</u> <u>de</u> <u>operare</u>, deci și cu alte calculatoare. *!!??!!* Da, nu este foarte credibil ceea ce afirm (recunosc că nici eu nu am crezut pînă nu am văzut!). *Adică asta înseamnă că programele de pe un IBM PC le putem rula pe un ATARI ST ?* Da, din punctul de vedere al utilizatorului, datorită emulatorului, putem lucra în MS-DOS ca pe un IBM compatibil, sau în CP/M ca pe un Junior sau Cub-Z, sau ca pe un MacIntosh. Bineînțeles nu există numai avantaje! Principalul dezavantaj este viteza de lucru. Față de un IBM PC original, programele MS-DOS sub emulator pe ATARI ST merg de 3 ori mai încet. Pe cine deranjează acest lucru ? Fiecare hotărește singur !

<u>Posibilități</u> <u>de</u> <u>dezvoltare</u> <u>(extindere)</u>

Din punct de vedere HARD în calculatorul ATARI ST există incorporate următoarele interfețe: floppy disk, hard disk, monitor, TV(opțional), MIDI In/MIDI Out, mouse, Joystick, RS232, paralelă, ROM cartridge. Există și un *user port* la care sînt scoase către exterior semnalele de care s-ar putea servi utilizatorul.

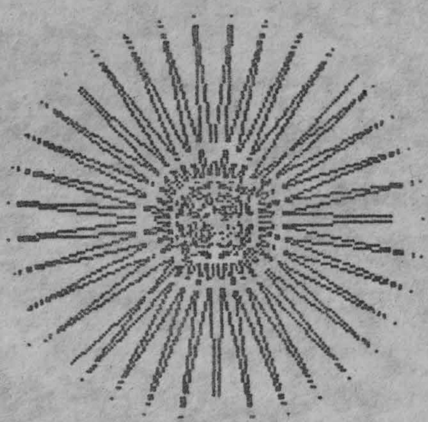
Din punct de vedere SOFT pentru calculatorul ATARI ST există implementări pentru limbajele de programare cele mai răspîndite ca BASIC, PASCAL, FORTRAN, C, PROLOG, FORTH, MODULA-2, ASSEMBLER, etc. In România există cel puțin următoarele:

<u>BASIC</u>	ST BASIC - interpretor
	GFA BASIC V2.0 - interpretor+compilator
	OMIKRON BASIC V3.0 - interpretor+compilator
<u>PASCAL</u>	PERSONAL PASCAL V1.05
	PASCAL ST PLUS V1.20
<u>FORTRAN</u>	PROSPERO FORTRAN 77
<u>C</u>	MEGAMAX C V1.1



VIRUSURILE CALCULATOARELOR

• ING. SÎRBU MIHAI •



Introducere

Ati auzit pina acum despre un calculator bolnav? Recent s-a relatat raspindirea unei epidemii provocate de un virus in calculatoarele din Pakistan. Publicatii cum ar fi PC World sau Wall Street Journal insereaza articole despre virusi electronici, vaccinuri si alte masuri de protectie. Se pare ca aici se ascunde o problema suficient de serioasa pentru a atrage atentia specialistilor in calculatoare.

Definitie si clasificare

Un **program virus** sau un **virus al calculatoarelor** (numit in continuare pe scurt virus) este o portiune de cod care, introdusa intr-un program executabil, il infecteaza. Lansat in lucru, codul respectiv identifica programe neinfectate la care ataseaza o copie a sa. S-a demonstrat ca, intr-un calculator care lucreaza normal, un virus se poate propaga in intervale de ordinul orelor. Odata generalizat, programul isi va activa ulterior operatiunea distructiva, conform modului specific in care a fost conceput.

Peter J. Denning, director la Research Institute for Advanced Computer Science, face o clasificare a virusilor in urmatoarele patru categorii:

1. **viermele** invadeaza o statie de lucru, pe care o scoate (partial sau total) din functiune.

2. **calul troian** este un program aparent util, dar care contine o secventa de cod ascunsa cu functii distructive. Secventa asteapta in general indeplinirea unor conditii prestabilite (spre exemplu data de 1 aprilie sau vineri in 13 ale lunii), cind se va activa singura.

3. **bacteria**, fara a se lasa mai prejos decit omologul ei biologic, se multiplica in avalansa. Ea lanseaza in executie nenumarate copii ale sale, si duce in final la ocuparea completa a capacitatii de prelucrare sau memorare a calculatorului gazda.

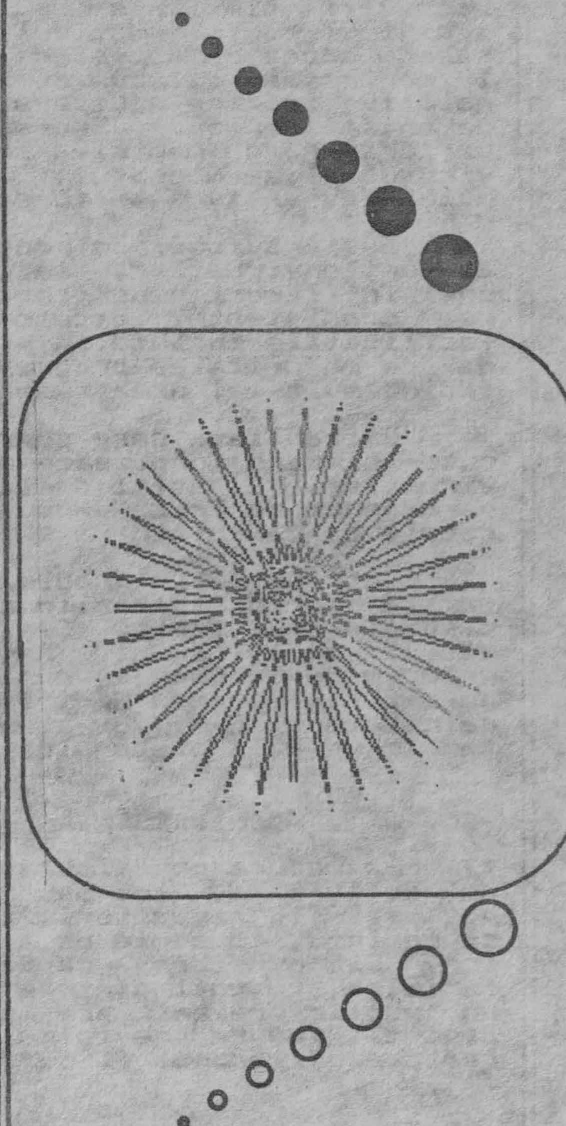
4. **virusul propriu-zis** produce in secret copii ale sale in codul masina al altor programe. Prin copierea fisierelor infectate pe alte sisteme se produce raspindirea lor, iar in cazul calculatorelor cuplate in retea se genereaza adevarate epidemii.

Exemple sugestive

La prima vedere, atacurile provocate de virusi si antidoturile lor par o sfidare intelectuala, insa, similar cu bolile umane, ele sint distructive si costisitoare. In acest sens vor fi prezentate citeva exemple.

Un mesaj aparent inocent a fost trimis din R.F.G. si s-a propagat prin reteaua BITNET a calculatoarelor I.B.M. din S.U.A. Acesta afisa un pom de iarna, iar copii au fost trimise tuturor celor prezenti pe listele de tranzactii comerciale ale calculatoarelor gazda! In scurt timp intreaga retea s-a blocat si a trebuit sa fie inchisa pina la eliminarea din sistem a tuturor exemplarelor.

Lucrind pe calculator, un student atent, a observat ca anumite programe de biblioteca devin mai mari fara nici un motiv aparent. A gasit segmentul de cod vinovat care, in zilele de vineri ce cadeau in data de 13 ale lunii, micsora viteza de lucru a calculatorului cu circa 80% . Codul urma sa distruga toate fisierele vineri, 13 mai 1988. Prin detectarea si stergerea tuturor copiilor virusului, s-a reusit inlaturarea pericolului.



Virusurile calculatoarelor

Masuri preventive

In fata valului mereu crescind de criminalitate informatica, autoritatile din S.U.A. au luat masuri severe. Pedepsele pot

ajunge la amenzi de 25.000 $ si pina la 5 ani de inchisoare. Vinovatii sint insa greu de gasit, si multe victime nu solicita sprijinul legii. Motivul? Publicitatea nedorita pentru firma pe care o aduce un astfel de caz. Ca urmare, utilizatorii au trecut la contramasuri, cum ar fi vaccinuri, anticorpi, imunizari. Desi relativ ieftine (citeva sute de dolari) ele nu dau insa o garantie deplina. Sanse mai mari le au cele care ridica si o bariera hard impotriva invadatorilor. Intr-o infruntare in care virusul alege momentul, locul si metoda atacului, sarcina sistemului desfensiv este foarte dificila.

Desi autorii de programe au pareri impartite cu privire la actiunile virusilor, majoritatea au luat masuri pentru evitarea unei infiltrari nedorite in sistemele realizate de ei. Verificarea consistentei datelor si a codului masina face parte dintre facilitatile incluse in programele dezvoltate recent. Se recomanda ca majoritatea programelor sa contina un astfel de segment, facind mult mai grea raspindirea unor virusi.

Exita liste care cuprind programele de tip cal troian cunoscute si care furnizeaza si diferite mijloace in lupta impotriva virusilor. O astfel de lista este "Dirty Dozen", actualizata in permanenta de Eric Newhouse, si care este disponibila in diverse retele de calculatoare.

Intr-un fel, problema este similara patologiei umane. Virusurile (prin eforturile autorilor) pot fi modificati sa ignore masurile existente. Cea mai buna aparare ramine igiena - ca de altfel si la oameni. Regula ar fi: sa nu folositi niciodata un program copiat sau imprumutat, daca nu sinteti absolut siguri ca nu este infectat. Nu-i lasati pe altii sa-si ruleze programe pe calculatorul dvs. Si atentie marita in toate cazurile de comportare ciudata a sistemului de calcul.

Concluzii

Pe masura ce calculatorul patrunde in cele mai multe domenii de activitate, tot mai multi oameni invata sa programeze. Unii pot sa fie rauvoitori, iar altii de-a dreptul criminali. In consecinta, o serie de autori doresc sa ascunda problema virusilor. Ei considera ca tacerea va limita raspindirea lor. Din fericire, nu toti sint de aceeasi parere. Ca si in cazul bolilor, un public prevenit are mai multe sanse in lupta contra raului. Discutii despre prevenire, identificare si cooperare in conceperea remediilor pot fi de un real folos in aceasta lupta.

Bibliografie

1. Levy, G.B., Computer Pathology, in International Laboratory, nov.1988.

2. A cure for the common virus, in Computer Buyer's Guide and Handbook, m/j 1988.

3. Betts, J., Shareware comes of age, in PC World, aug. 1988.

4. Campbell, G., Antivirus Vaccine, in PC World, aug.1988.



TEMA DE CASA

SE DA: UN TELEVIZOR CARE POATE RECEPTIONA DOUA PROGRAME TV, UN CALCULATOR SI CEVA HARD MANUFACTURAT.
SE CERE: SA SE AFISEZE PE TELEVIZOR, CELALALT PROGRAM TV, INTR-UN COLT; LA SCHIMBAREA PROGRAMULUI, SA SE COMUTE INTRE ELE CELE DOUA IMAGINI: IN COLT CEEA CE A FOST PE TOT ECRANUL SI VICEVERSA; AFISAJUL DIN COLT ESTE 1/100 DIN TOATA IMAGINEA TV, ASTFEL: SE MEMOREAZA TOT A ZECEA LINIE DIN IMAGINE, SI DIN FIECARE LINIE, TOT AL ZECELEA PUNCT; REZULTA O MATRICE DE 650/10 LINII SI 866/10 COLOANE, FIECARE ELEMENT AL MATRICII FIIND UN PUNCT DIN CELE 650*866 PUNCTE ALE INTREGII IMAGINI; AFISAREA UNUI CADRRU SE FACE IN TIMPUL MEMORARII CADRULUI URMATOR; REZULTA O FRECVENTA A CADRELOR DIN IMAGINEA DIN COLT DE 1/2 DIN 25 DE CADRE CITE ARE O IMAGINE NORMALA. (VARIANTA, POATE NU CEA MAI BUNA). SUCCES !

MINISTERUL

EDUCATIEI SI

INVĂȚĂMINTULUI

Casa Universitarilor

Timișoara

Buletin al

Clubului Programatorilor

INF

Nr. 1/1989

COLECTIVUL DE REDACTIE:

conf. dr. ing. CRIȘAN STRUGARU
s.l.dr. ing. ȘTEFAN HOLBAN
s.l. dr. ing. IONEL JIAN
ing. CONSTANTIN COZMIUC

TEHNOREDACTAREA

EWELINE BELMUSTAȚĂ
CRISTIAN BÎRLONCEA

INF

NR 2/1989